目錄

前端开发者手册

这是任何人都可以用来学习前端的实践手册, 它概述并讨论了前端工程的实践: 该如何学习以及实践时该使用什么工具.

撰写该手册的目的有两个: 一是为潜在以及正在实践的前端开发人员提供一个包括学习资料和开发工具的专业资源; 二是该手册可以被管理者, CTO, 讲师和猎头用来作为洞察前端开发的实践.

该手册的内容支持Web技术(HTML, CSS, DOM, 和 JavaScript), 并且手册提供的解决方案都直接建立在这些开放的技术之上. 手册中所引用的素材和讨论都是最好的或者当前前端开发者们需要面对的问题.

该手册不应该被视为一个前端开发者对所有可用资源的综合大纲, 其价值在于简洁, 专注和及时管理足够的分类信息, 不致于任何人沉浸在任何一个特定的主题.

该手册会每年发布一个更新内容.

该手册分为三个部分:

第一部分：前端开发实践

第一部分会大致描述前端工程的实践

第二部分：学习前端开发

第二部分为成为一个前端开发人员确定了自主学习的直接资源

第三部分：前端开发工具

第三部分会简单地讨论一些前端开发工具的使用

在线阅读: 前端开发者手册

资源下载: pdf mobi

Issues/Suggestions/Fixes: Front-end Developer Handbook

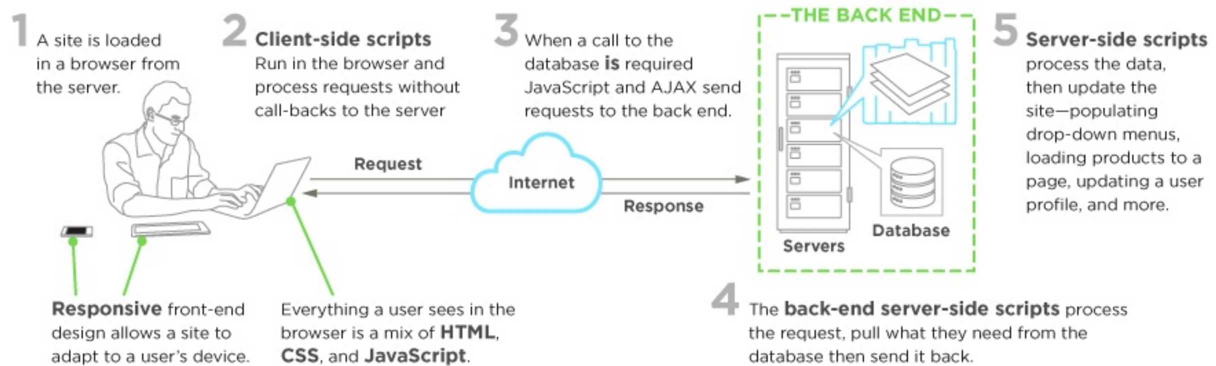
说明: 该手册参考 Front-end Developer Handbook 电子书所译, 不当之处, 欢迎 pr 或提出 issue.

联系译者:

- 博客: Blog
- 微博: 会飞的Pikachu
- QQ交流群: 加入QQ群

什么是前端开发者?

一个前端开发者, 要会使用Web技术(如：HTML,CSS,DOM和JavaScript)设计和开发网站应用. 网站应用, 或运行于 Web平台 之上, 或用于编译非Web平台环境的输入(如：NativeScript).



图片来源：*https://www.upwork.com/hiring/development/front-end-developer/*

一般而言, 一个人可以通过学习 HTML,CSS,JavaScript进入前端开发领域, 这些代码运行在 Web浏览器, 无壳为浏览器, Web视图之中, 或用于编译本地运行环境的输入. 后文将详细介绍这四个运行场景.

Web浏览器是用于检索, 呈现和遍历万维网(WWW)信息的软件. 一般而言, 浏览器可以运行在台式机, 笔记本电脑, 平板电脑或手机. 但是近来, 几乎在任何事物上都能够发现浏览器(如: 冰箱上, 汽车里等).

最普遍的Web浏览器如下:

- Chrome
- Internet Explorer
- Firefox
- Safari

无壳浏览器是指没有图形用户界面的Web浏览器, 可以通过命令行接口控制达到网页自动化的目的(如: 功能测试, 单元测试等). 把无壳浏览器当做可以从命令行运行的浏览器, 它依然可以检索和遍历网页.

最普遍的无壳浏览器如下:

- PhantomJS
- slimerjs
- trifleJS

Webviews 被本地 OS 用来运行网页. 把Web视图当做Web浏览器中的iframe或者单个的Tab, 其嵌入于运行在设备上的本地应用程序中(如：iOS, android, windows).

Web视图开发最普遍的解决方案如下:

- Cordova (用于本地手机/平板应用)
- NW.js (即 Node-Webkit, 用于桌面应用)
- Electron (用于桌面应用)

最后, 前端开发者从 Web 浏览器开发环境中学到的东西也可以用于不受浏览器引擎驱动的环境下. 目前, 脱离 Web 引擎，使用 Web 技术(如: CSS 和 JavaScript)去创建真正的本地应用的开发环境正在出现.

此类环境的示例如下:

- NativeScript
- React Native

译者补充**:**

- 达到什么样的标准才能是大公司要的前端
- 写给初学前端工程师的一封信
- 前端路上的旅行
- React Native专题

第一部分：前端开发实践

第一部分会大致描述前端工程的实践

前端的工作职称

下面是一个前端开发者在职业发展中各种职称的描述列表. 对于前端开发者最普遍的职称是 "前端开发者" 或者 "前端工程师", 可以根据任何包含 "前端", "客户端", "web UI", "CSS", "HTML" 和 "JavaScript" 的职称推断一个人对 HTML, CSS 和 JavaScript 的了解程度.

前端开发者**/**工程师 (又称作 Web前端开发者/工程师, 客户端开发者/工程师, 前端软件开发者/工程师 或 UI 工程师)

这是通用的职称, 用于描述一个开发者对 HTML, CSS, JavaScript 有很熟练的掌握, 并能在Web平台上应用这些技术.

CSS/HTML开发者

这个职称用于描述一个开发者精通于 CSS 和 HTML, 但是对 JavaScript 和应用不熟悉.

前端**JavaScript(**可选**,** 应用程序**)**开发人员

当职称中包含 "JavaScript应用程序" 时, 这就表示此开发人员是一个拥有高级编程, 软件开发和应用程序开发技能(如: 有构建前端应用程序的实践经验)的高级 JavaScript 开发者.

前端**Web**设计师

当职称中包含 "设计师" 时, 这就表示此设计师不仅拥有前端技能(如: HTML & CSS), 还拥有专业的设计技能(视觉设计和交互设计).

Web/前端用户界面**(**又称 **UI)**开发者**/**工程师

当职称中包含 "界面" 或者 "UI" 时, 这就表示此开发人员除了掌握前端技能, 还拥有交互设计技能.

移动**/**平板前端开发者

当职称中包含 "移动" 或者 "平板" 时, 这就表示此开发人员在开发运行在移动或平板设备上的前端应用(或本机程序, 或运行在Web平台, 例如浏览器)很有经验.

前端 **SEO** 专家

当职称中包含 "SEO" 时, 这就表示此开发者对一个 SEO 策略定制前端技术有着丰富的经验.

前端可访问性专家

当职称中包含 "可访问性" 时, 这就表示此开发者对定制支持可访问性要求和标准的前端技术有着丰富的经验.

前端开发运维

当职称中包含 "开发运维" 时, 这就表示此开发者对软件开发实践与合作, 集成, 部署, 自动化和测量有着丰富的经验.

前端测试**/QA**

当职称中包含 "测试" 或 "QA" 时, 这就表示此开发者对测试和软件管理, 包括单元测试, 功能测试, 用户测试和 A/B 测试有着丰富的经验.

注意, 若你在职称中碰到 "全栈" 或 类似于 "Web开发者" 一样的职位, 这些可能是雇主用来描述这个角色负责 Web/应用程序 开发的各个方面, 即包括前端和后端(还可能包括设计).

译者补充:

- 谈前端工程师的职业规划

前端开发者常用的网络技术



前端开发者常用的网络技术如下:

- Hyper Text Markup Language(超文本标记语言, 又称 HTML)
- Cascading Style Sheets (层叠式样式表, 又称 CSS)
- Document Object Model (文档对象模型, 又称 DOM)
- JavaScript Programming Language (JavaScript编程语言, 又称: ECMAScript 6, ES6, JavaScript 2015)
- Web API's (Web应用程序接口, 又称 HTML5 API 或浏览器 API)
- Hypertext Transfer Protocol (超文本传输协议, 又称 HTTP)
- Uniform Resource Locator's (统一资源定位符, 又称 URL)
- JavaScript Object Notation (JavaScript对象表示, 又称 JSON)
- Web Content Accessibility Guidelines (网页内容无障碍设计指南, 又称 WCAG) & Accessible Rich Internet Applications (富Internet应用程序的无障碍设计, 又称 ARIA)

根据相关的文档和规范, 这些技术定义如下. 作为一个比较, 你可以在 platform.html5.org 上看到所有与Web相关的规范.

Hyper Text Markup Language(超文本标记语言, 又称 HTML)

> 超文本标记语言, 通常被称为 HTML, 是被用于创建网页的标准标记语言. Web浏览器能将 HTML 文件渲染成可见的或者可听到的. HTML 随着线索提示, 语义化地描述了网站的结构, 使它成为一种标记语言, 而不是编程语言. - wikipedia.org

更多相关的文档/规范:

- W3C HTML5 规范 : HTML5 是WWW核心语言的主要修订
- HTML 元素在线标准

- HTML 在线语法
- 所有 W3C HTML 规范
- HTML 元素参考
- HTML 属性参考
- 全局属性

Cascading Style Sheets (层叠式样式表, 又称 CSS)

> 层叠式样式表(CSS)是用于描述外观和格式化标记语言编写的文档的样式表语言. 尽管经常被用来改变用 HTML 和 XHTML 编写的网页和用户界面的样式, 但也可用于任何 XML 文档, 包括纯 XML, SVG 和 XUL. 跟 JavaScript 和 HTML 一样, CSS是被大多数网站用于为Web应用程序创建富有吸引力的网页, 用户界面的一种基础技术, 也为许多移动应用程序创建用户界面. - wikipedia.org

更多相关的文档/规范:

- CSS 2.2规范
- CSS 3选择器
- 所有 W3C CSS 规范
- CSS 参考

Document Object Model (文档对象模型, 又称 DOM)

> 文档对象模型用于代表和对象交互的HTML, XHTML 和 XML 文档, 是一种跨平台和语言无关性的约定. 每一份文档的所有节点被组织成一种树结构, 称为 DOM 树. DOM 对象通过使用对象上的方法被处理和操作, 一个 DOM 对象的公共接口被指定为它的应用程序编程接口(API). - wikipedia.org

更多相关的文档/规范:

- W3C DOM4
- DOM 在线标准
- DOM 3 事件规范

JavaScript Programming Language (JavaScript编程语言, 又称: ECMAScript 6, ES6, JavaScript 2015)

> JavaScript 是一种高级的, 动态的, 无类型的和解释型的编程语言, 它已经在 ECMAScript 语言规范中被标准化. 跟 HTML 和 CSS 一样, JavaScript 是 WWW 内容生成的第三种必不可少的技术; 大多数的网会使用Javascript, 并且 Javascript 被所有现在Web浏览器支持. JavaScript 基于原型和函数优先的特点, 使它成为多范型的语言, 支持面向对象的, 命令式的, 和函数式编程风格. JavaScript 能提供 API 来处理文本, 数组, 日期和正则表达式, 但不包括任何 I/O, 如网络, 存储或图形工具, 对这些的依赖取决于宿主环境中嵌入了什么. - wikipedia.org

更多相关的文档/规范:

- ECMAScript 2015 语言规范

Web API's (Web应用程序接口, 又称 HTML5 API)

> 当使用 JavaScript 为Web程序写代码时, 有很多不错的 API 是可以利用的. 下面列举了所有在Web APP 或网站开发中可能会用到的接口. - Mozilla

更多相关文档:

- Web API 接口

Hypertext Transfer Protocol (超文本传输协议, 又称 HTTP)

> 超文本传输协议是一个用于分布式, 协作和超媒体信息系统的应用协议, 是 WWW 数据通信的基础. - wikipedia.org

更多相关规范:

- Hypertext Transfer Protocol -- HTTP/1.1
- Hypertext Transfer Protocol version 2 draft-ietf-httpbis-http2-16

Uniform Resource Locator's (统一资源定位符, 又称 URL)

> 一个 URL (也称Web地址)是一个资源引用, 指定了资源在计算机网络和检索机制中的位置. 与之类似的概念是 Uniform Resource Identifier(URI), 尽管许多人认为两个术语可以互换使用, 但 URL 是统一资源标识符(URI)[3] 的具体类型. 一个 URL 意味着一种访问资源量的方式, 但这并不适用于 URI. [4][5]URL 不仅常用于引用一个网页(http), 也可用于文件传输(ftp), 邮件(mailto), 数据库访问(JDBC) 和许多其它应用. - wikipedia.org

更多相关规范:

- URL
- URL 在线标准

JavaScript Object Notation (JavaScript对象表示, 又称 JSON)

> JSON, 有时也称 JavaScript 对象表示, 是一种使用人类可读的文本传输由键值对组成的数据对象的开放格式. 对于异步浏览器/服务器通信(AJAJ), JSON 是主要的数据格式, 很大程度上代替了 XML(AJAX). 尽管最初是从 JavaScript 脚本语言衍生而来, 但是 JSON 是语言无关性的数据格式, 在许多编程语言中, 代码解析和生成 JSON 是很容易的. JSON 的格式最初是由 Douglas Crockford 指定的, 但目前却被描述成两种标准: RFC 7159 和 ECMA-404. ECMA 标准只允许被合法的语法语句描述, 而 RFC 则提供了一些语义化描述和安全考虑. JSON 的官方网络媒体类型 application / JSON, 扩展名是 .json. - wikipedia.org

更多相关规范:

- JSON 介绍
- JSON 数据交换格式
- JSON API

Web Content Accessibility Guidelines (网页内容无障碍设计指南, 又称 WCAG) & Accessible Rich Internet Applications (富Internet应用程序的无障碍设计, 又称 ARIA)

> 无障碍设计是指产品, 设备, 服务, 或者环境是为残疾人设计的. 无障碍设计的概念意味着与一个人的辅助技术(例如, 电脑屏幕阅读器)相兼容, 确保直接访问(即独立)和"间接访问".
> - wikipedia.org

- 无障碍设计网络倡议 (WAI)
- 网页内容无障碍设计指南 (WCAG) 的目前状态
- 富 Internet 应用程序的无障碍设计 (WAI-ARIA) 的目前状态

译者补充:

- Web 端开发常用资源
- Web 前端开发资源

前端开发的技术栈



对于任何类型的前端开发人员, HTML, CSS, DOM, JavaScript, HTTP/URL 和浏览器利用是基本的技术要求.

对于HTML, CSS, DOM, JavaScript, HTTP/URL 和浏览器开发之外的, 一个前端开发者还应该掌握下面技术列表中的一个或多个:

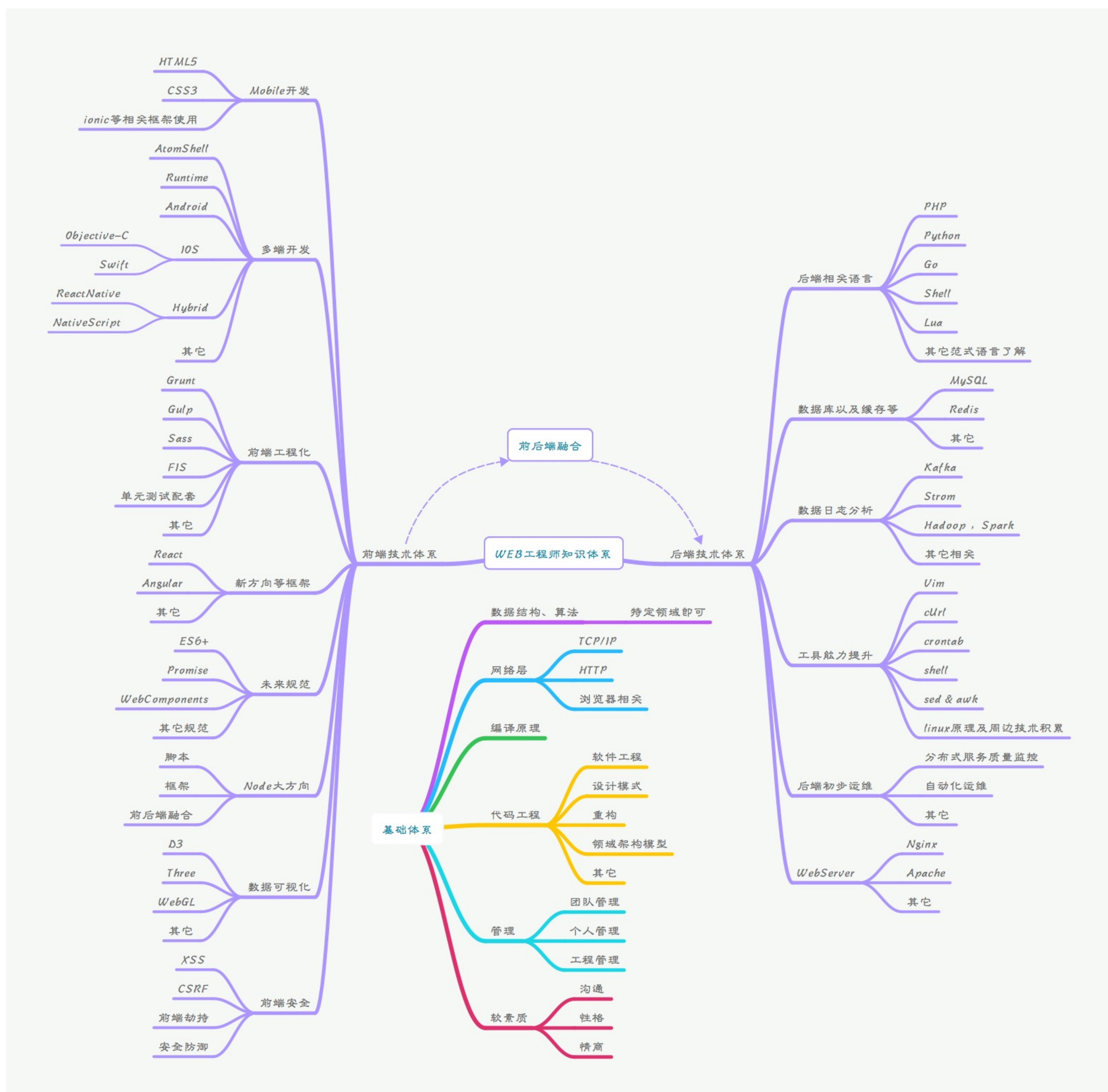
- Content Management System (内容管理系统, 又称 CMS)
- Node.js
- Cross-browser testing (跨浏览器测试)
- Cross-platform testing (跨平台测试)
- Unit Testing (单元测试)
- Cross-device testing (跨设备测试)
- Accessibility / WAI-ARIA (无障碍访问/无障碍富Internet应用程序)
- Search Engine Optimization (搜索引擎优化, 又称 SEO)
- Interaction or User Interface design (交互或用户设计)
- User Experience (用户体验)
- Usability (可用性/易用性)
- E-commerce Systems (电子商务系统)
- Portal Systems (门户系统)
- Wireframing (框架)
- CSS layout / Grids (CSS 布局/栅格系统)
- DOM manipulation (e.g. jQuery) (DOM 操作)
- Mobile Web Performance (移动Web性能)
- Load Testing (载荷测试)
- Performance Testing (性能测试)

- Progressive Enhancement / Graceful Degradation (渐进增强/优雅降级)
- Version Control (e.g. GIT) (版本控制)
- MVC / MVVM / MV *(MV* 框架)
- Functional Programming (函数式编程)
- Data Formats (e.g. JSON, XML) (数据格式)
- Data API's (e.g Restful API) (数据API)
- Web Font Embedding (Web字体嵌入)
- Scalable Vector Graphics (可伸缩向量图形, 又称 SVG)
- Regular Expressions (正则表达式)
- Content Strategy (内容策略)
- Microdata / Microformats (微数据/微格式)
- Task Runners, Build Tools, Process Automation Tools (任务管道, 构建工具, 过程自动化工具)
- Responsive Web Design (响应式设计)
- Object Oriented Programming (面向对象编程)
- Application Architecture (应用架构)
- Modules (模块)
- Dependency Managers (依赖关系管理)
- Package Managers (包管理)
- JavaScript Animation (JavaScript 动画)
- CSS Animation (CSS 动画)
- Charts / Graphs (图表/图形)
- UI widgets (UI工具集)
- Code Quality Testing (代码质量测试)
- Code Coverage Testing (代码覆盖测试)
- Code Complexity Analysis (代码复杂度测试)
- Integration Testing (集成测试)
- Command Line / CLI (命令行/命令行界面)
- Templating Strategies (模板策略)
- Templating Engines (模板引擎)
- Single Page Applications (单页应用)
- XHR Requests (aka AJAX) (XHR 请求, 又称 AJAX)
- Web/Browser Security (Web/浏览器安全)
- HTML Semantics (HTML 语义化)
- Browser Developer Tools (浏览器开发者工具)

译者补充:

- 前端技能汇总
- WebFrontEndStack


前端工程师
Web服务
YQL/Company Open Platform
APP Engine/Cloud
Google Open API/Github
前沿技术社区/会议
YDN/YUIConf/JSConf
QCon/Velocity
D2/WebRebuild
NodeParty/HTML5研究小组
UED Teams
移动终端
jQueryMobile/html5/css3
iPhone/iPad/iOs/android
responsive UI Design
安全
CSRF/XSS
ADsafe/Caja/FBJS/Sandbox
软技能
知识管理/总结分享
沟通技巧/团队开发/需求管理/PM
代码模块化开发/代码版本管理
交互设计/可用性/可访问性知识
开发流程/部署
JSLint/CSSLint/YUICompressor
JSMin/TPacker-minifier
Ant/Make
JSDoc/YUIDoc
LAMP
前端框架/库
jQuery/YUI2/YUI3
Prototype/Mootools/ExtJS
Smarty/Django/Zend
YUITest/QUnit/Jasmine
前端测试/前端MVC
浏览器兼容性
IE6/7/8/9
Firefox3.6/4/5/6/7/8...
Chrome/Opera/Safari
切页面
HTML
CSS
PhotoShop
编程语言
JavaScript
PHP/Python/Perl/Ruby/BashShell...
NodeJS/FlashAS
调试工具
Firebug/Firebug-lite/Web Inspector
YSlow/Smushit
IEDeveloperToolBar/IETester
SuperPreview/JsBeautifier
Fiddler/WireShark/tcpdump
开发工具
VIM/Aptana/Notepad++/EditPlus...
SygWin/svn/git
计算机知识储备
编译原理/计算机网络
操作系统/算法原理
软件工程/软件测试原理
前端标准/规范
w3c/DOM/BOM/XHTML/XML/JSON/JSONP
HTML5/CSS3
ECMAScript3/ECMAScript5
CommonJS/AMD
HTTP1.1
性能
WebPageTest
ShowSlow/YSlow/34Rule
PageSpeed
HttpWatch
DynaTrace's Ajax
编程知识储备
数据结构/MVC
面向对象/切面编程
函数式编程
JavaScript设计模式
JavaScript编程模式


WEB工程师知识体系
前后端融合
前端技术体系
后端技术体系
Mobile开发
HTML5
CSS3
ionic等相关框架使用
多端开发
AtomShell
Runtime
Android
IOS
Objective-C
Swift
Hybrid
ReactNative
NativeScript
其它
前端工程化
Grunt
Gulp
Sass
FIS
单元测试配套
其它
新方向等框架
React
Angular
其它
未来规范
ES6+
Promise
WebComponents
其它规范
Node大方向
脚本
框架
前后端融合
数据可视化
D3
Three
WebGL
其它
前端安全
XSS
CSRF
前端劫持
安全防御
后端相关语言
PHP
Python
Go
Shell
Lua
其它范式语言了解
数据库以及缓存等
MySQL
Redis
其它
数据日志分析
Kafka
Strom
Hadoop，Spark
其它相关
工具能力提升
Vim
cUrl
crontab
shell
sed & awk
linux原理及周边技术积累
后端初步运维
分布式服务质量监控
自动化运维
其它
WebServer
Nginx
Apache
其它
基础体系
数据结构、算法
特定领域即可
网络层
TCP/IP
HTTP
浏览器相关
编译原理
代码工程
软件工程
设计模式
重构
领域架构模型
其它
管理
团队管理
个人管理
工程管理
软素质
沟通
性格
情商

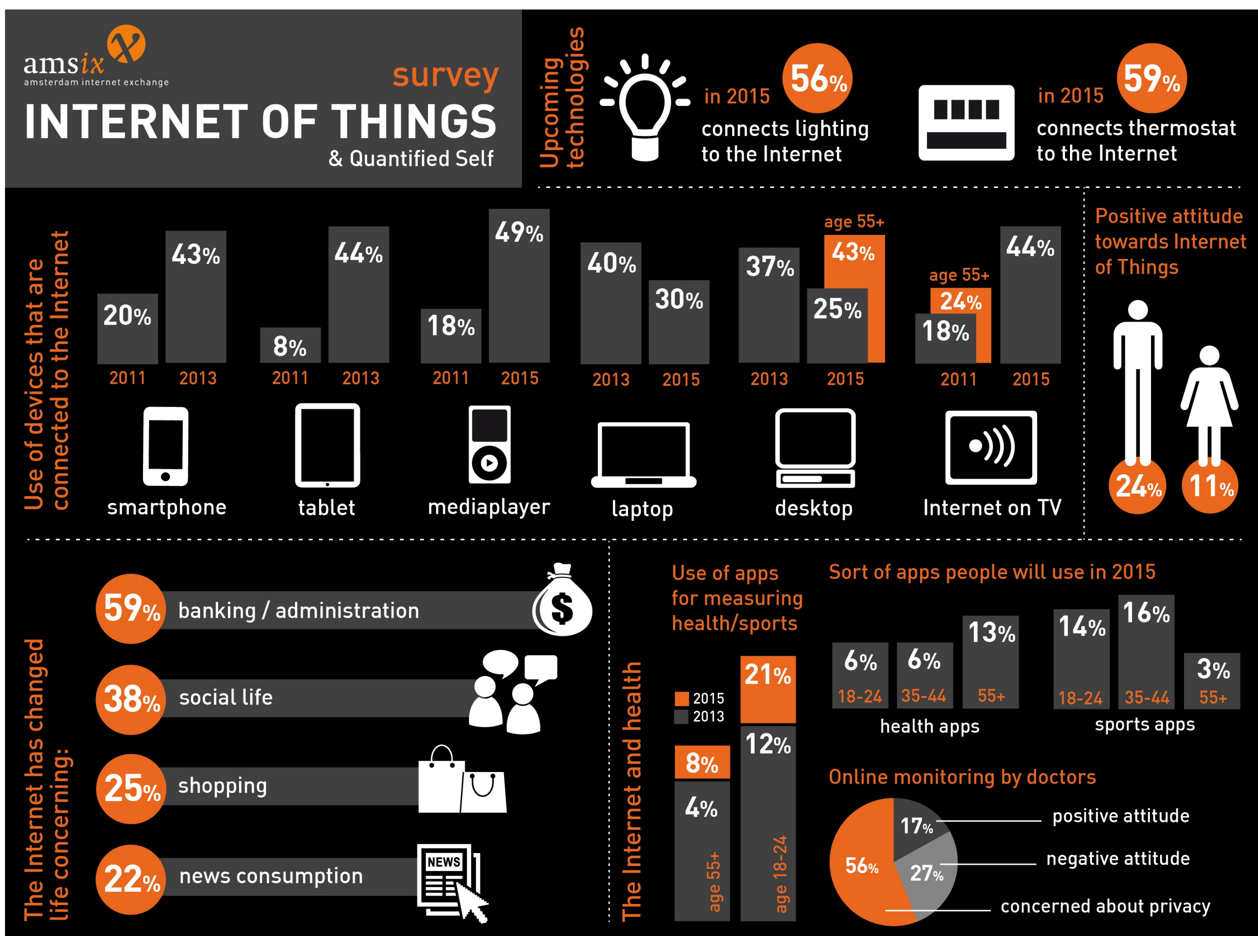
前端开发做什么

一个前端开发者能在下面的操作系统列(又称: OS)表中之一上手写运行在Web平台(如: 浏览器)之上的 HTML, CSS 和 JS:

- Windows
- Windows Phone
- OSX
- iOS
- Android
- Ubuntu (or Linux)
- Chromium

操作系统运行在下面中的一个或者多个设备之上:

- Desktop computer
- Laptop / Netbook computer
- Mobile phone
- Tablet
- TV
- Watch
- Things (任何你能想到的, 汽车, 冰箱, 灯光, 温控器等)



一般来说，前端技术通过使用下列运行时场景，能运行在前面提到的操作系统和设备之上：

- 运行在 OS 上的Web浏览器(如: Chrome, IE, Safari, Firefox)
- 运行在 OS 上并由 CLI 驱动的 headless浏览器(如: plantomJS)
- 一个Web视图/嵌入本机程序的浏览器Tab(当做 iframe)作为运行时环境，作为与本机 API 通信的桥梁. 典型的Web视图应用包括一个由Web技术(HTML, CSS, 和 JS)构建的 UI.(如: Apache Cordova, NW.js, Electron)
- 一个由Web技术创建的本机程序会在运行时作为与本机 API 通信的桥梁，被解释执行，UI 将使用本机的UI部分(如: IOS 本机控制)而不是Web技术控制(NativeScript, React Native).

团队中的前端

在一个设计和开发 Web网站, Web应用, 或者基于Web技术的本机应用的团队, 前端开发者是典型的只有一个人. (注意: 构建一切的开发者曾经被称为"Web网管", 但这些罕见的和神秘的开发人员现在被称为"全栈开发者").

一个构建专业的Web网站或软件的最小化团队也应该包含下列角色:

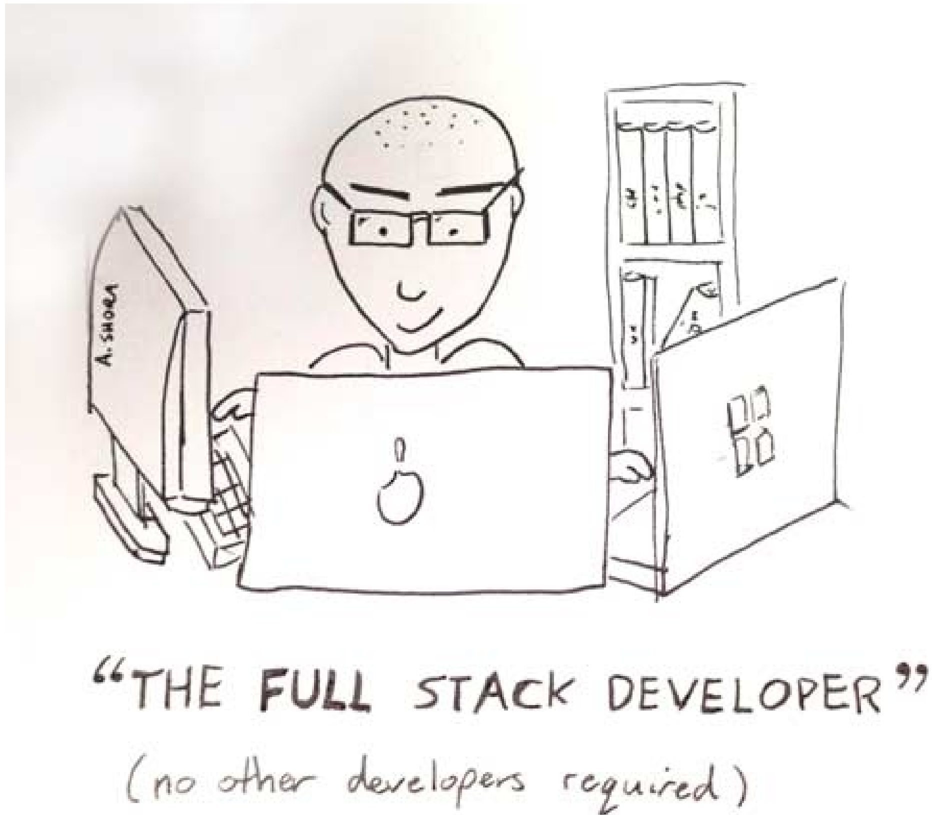
- 视觉设计 (字体, 颜色, 字距, 视觉概念&主题)
- UI/交互设计/信息架构师 (制定框架, 指定所有用户交互, UI功能和结构信息)
- 前端开发者 (写能够在客户端/设备上运行的代码)
- 后端开发者 (写能够在服务端运行的代码)

这些角色是根据技能的覆盖排序的(后面的角色的技能会覆盖前面的). 一个前端开发者很擅长处理 UI/交互设计, 后端开发者也一样. 团队成员承担多个角色是很少见的.

一个大的团队可能包含下列角色, 而不是如上面所展示的:

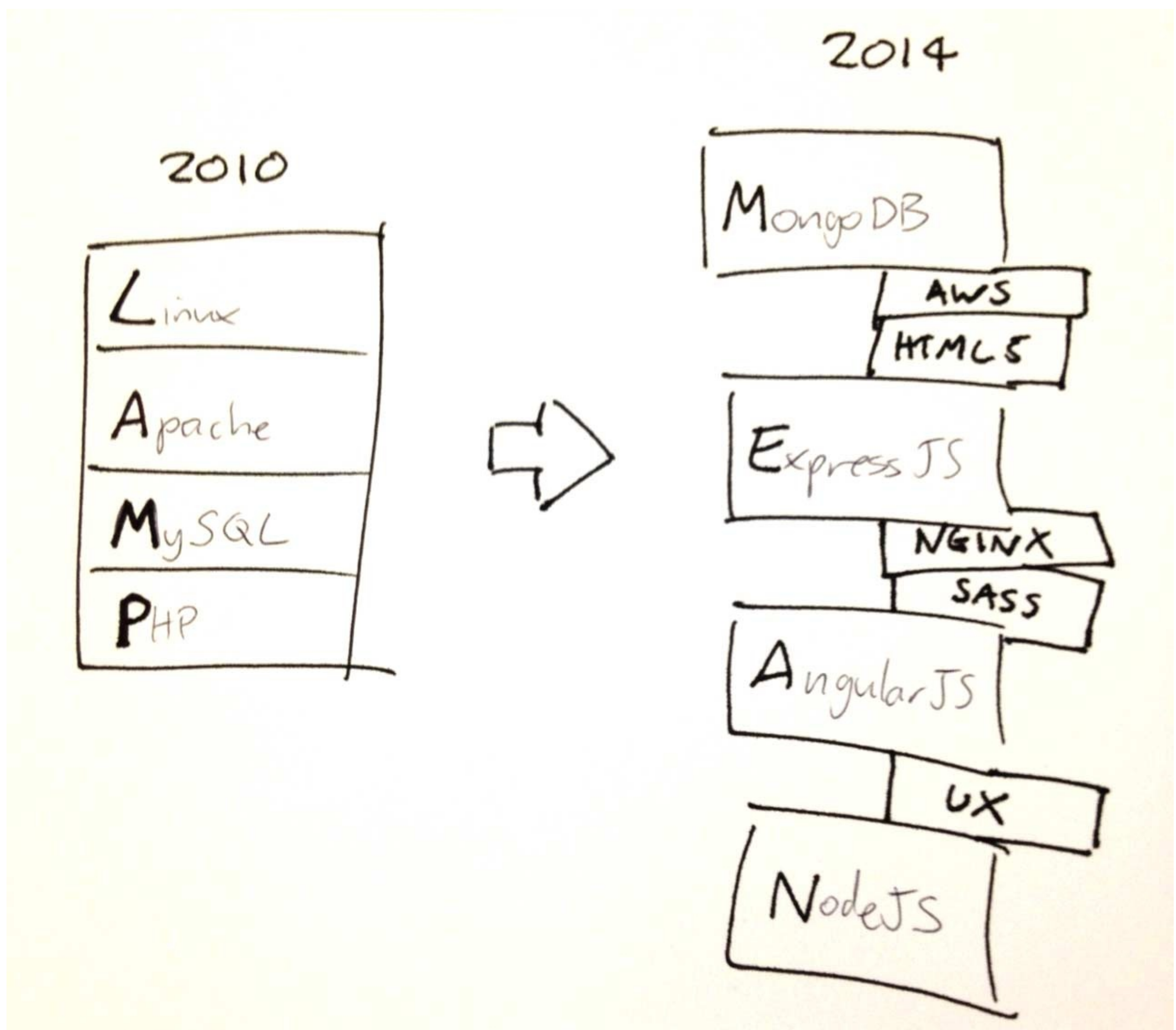
- 视觉设计
- UI/交互设计/信息架构师
- **SEO** 策略师
- 前端开发者
- 开发**-**运维工程师
- 后端开发者
- **API** 开发者
- 数据库管理员
- **QA** 工程师**/**测试人员

全才神话



这是个需要设计和开发一个web解决方案的角色, 不仅需要有很深厚的技术栈, 而且需要在视觉设计, UI/交互设计, 前端开发和后端开发有大量的经验. 任何可以以一个专业的水平承担这 4 个角色中的一个或多个的人(又称全才或全栈开发者)是很少的.

务实一点, 你应该寻求, 或者雇佣一个在其中一个角色堪称专家的人. 那些声称在一个或多个角色上是专家的人是非常罕见的,


2010
Linux
Apache
MySQL
PHP
2014
MongoDB
AWS
HTML5
ExpressJS
NGINX
SASS
AngularJS
UX
NodeJS

前端面试

你可能被问到的问题:

- 前端工作面试问题
- 前端开发面试问题
- 每个 JavaScript 开发者应该知道的 10 个面试问题
- 前端测验
- JavaScript 测验

你可以问的问题:

- 一个开源的开发人员可以向潜在雇主提问的问题列表

译者补充:

- 前端开发面试题大收集
- 前端开发面试问题及答案整理
- 收集的前端面试题和答案
- 写给前端面试者
- Awesome Interviews

前端工作版块

可以列出很多寻找技术工作的方法. 下面的有限列表是和寻找一份具体的前端工作最相关的资源:

- frontenddeveloperjob.com
- authenticjobs.com
- weworkremotely.com
- jobs.github.com
- careers.stackoverflow.com
- angularjobs.com
- jobs.emberjs.com
- jobs.jsninja.com
- css-tricks.com/jobs
- glassdoor.com

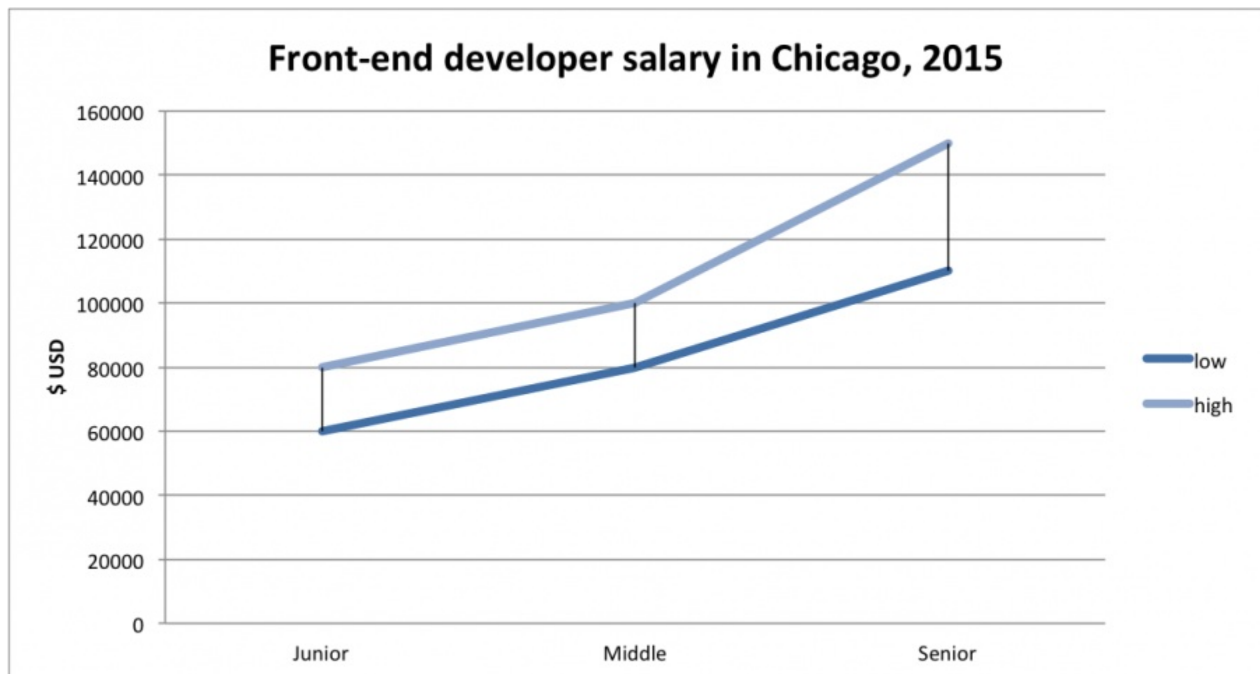
译者补充:

- w3ctech
- w3cfuns
- 内推网
- 拉勾网
- 前端网
- CSDN
- 竞鹿网
- 100Offer
- SegmentFault
- 周伯通
- 内聘网
- 博客园
- V2EX
- 简历模板
- 面试问题1 面试问题2

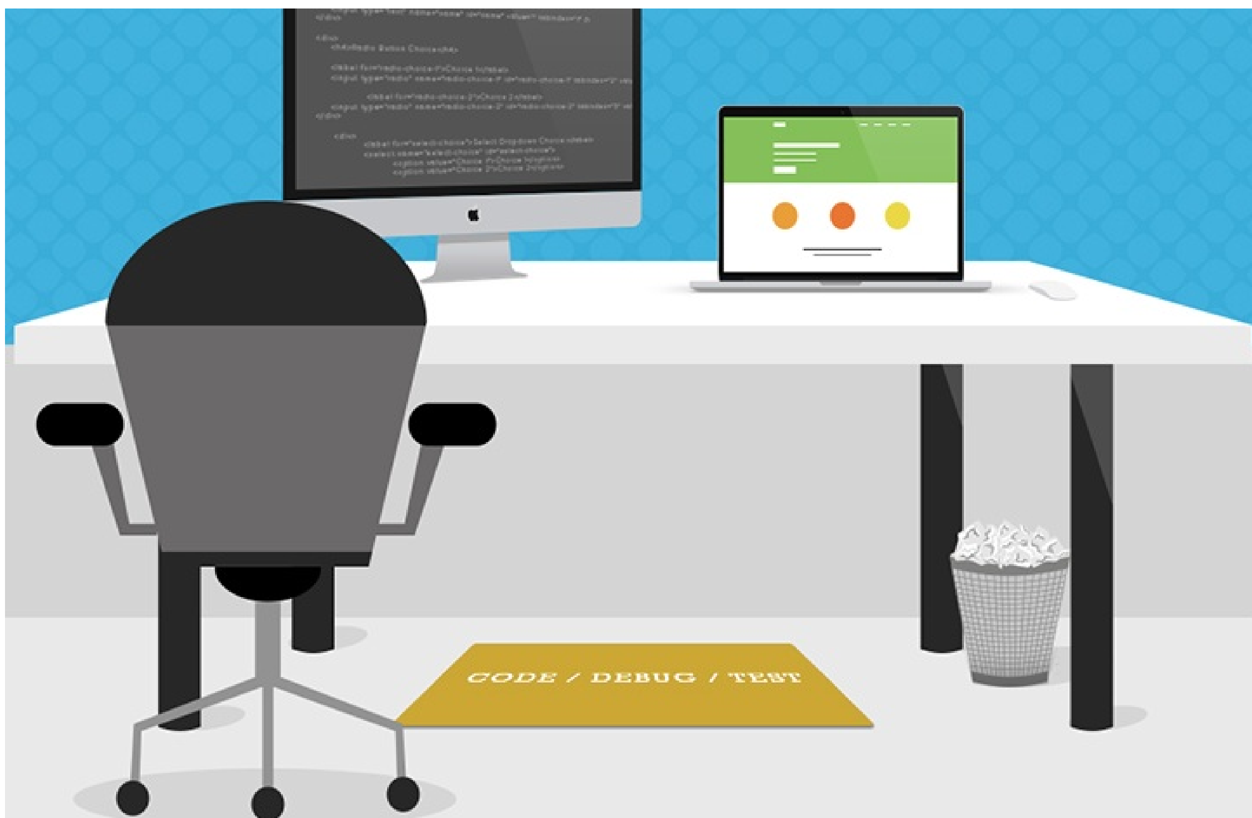
前端薪资

在美国, 前端开发者的平均薪资是 $75K

一个有经验的前端开发者可以去任何想去的地方生活(远程工作), 并且一年能赚的钱超过 $150k(访问 angel.co, 注册, 查找超过 $150k 的前端工作).



如何成为前端开发者?



那么, 怎么才能成为一个前端开发者呢? 这个问题很复杂, 因为直到现在, 你也不能去一所大学获得前端工程师的学位, 并且我也很少听说 或者遇见通过编写专业地 HTML, CSS 和 JavaScript 来获得一个无用的计算机科学学位或平面设计学位. 事实上, 现在的大部分前端 开发者都是通过自学成为开发者和没有经过传统训练的程序员. 为什么会是这种情况呢?

前端开发人员不是一个视觉设计师或一个交互设计师, 设计学校不是磨练前端技能的地方; 前端开发者也不是一个受过传统教育地计算机科 学研究生, 传统教育并不专注于让一个人为前端开发做准备. 实际上, 在美国的高等教育系统中(例如大学), 紧跟传统教学方式可能会阻碍 一个人置身实践, 而前端开发最需要实际经验. 在今天, 如果你想成为一个前端开发者, 你可以自学或者参加一些不被认可的项目, 课程, 训练营和班级.

前端工程师会精巧地创建用户界面依赖的骨架. 有时, 他们要足够关注交互设计, 因为他们会编写 UI 交互的底层代码. 因此, 现在的许多 实践是前端工程师使用编程技巧达不到的, 但是, 从另一个方面来说, 和其它类型程序员转前端开发相比, 似乎有更多的设计师转前端开发者. 当然, 由于 JavaScript 已经成熟, 更多的受过传统教育的程序员愿意将他们的知识带到前端实践中. 你可能没有意识到前端开发人员并不 总是被认为是"真正的"程序员, 但是, 时代正待正在改变.

正如所有人说的那样, 我相信作为前端开发人员, 职业生涯道路是一个未知的过程. 我能说的是, 要成为一名前端工程师, 就必须知道和在一 个高层次的水平上使用 HTML, CSS 和 JavaScript, 也不会忽略交互设计或者传统编程所应该知道的技能. 实际上, 从我的经验来看, 最

好的前端开发者通常会掌握交互设计和基于 Web 平台(例如浏览器, HTML, CSS, DOM 和 JavaScript)的编程. 不管出于什么原因, 还有 很多关于前端开发的知识往往没被发现, 也就是说, 前端工程更像一些由自学的人组成的实践, 而不是一个直接对应有组织和认可的高等教育的教 学重心的领域.

如果我从现在开始决定成为一名前端开发人员, 我会努力按照下面所概括的过程进行学习. 学习过程中, 我会假设你是自己最好的老师.

1. 粗略了解 Web 是怎么工作的, 确保你知道域名, DNS, URL, HTTP, 网络, 浏览器, 服务器/服务托管, 数据库, JSON, API, HTML, CSS DOM 和 JavaScript. 了解这些的目的是确保你知道它们如何一起工作以及每部分用于做什么. 专注于高水平的前端架构概述. 从简单的网页 制作开始, 并简单学习一下 本机 Web 应用(又称 SPA).
2. 学习 HTML, CSS, 可访问性和 SEO.
3. 学习 UI 设计模式的基本原理, 交互设计, 用户体验设计和可用性
4. 学习编程的基本原理
5. 学习 JavaScript
6. 学习 JSON 和 API
7. 学习 CLI/命令行
8. 学习软件工程实践(如: 应用设计/架构, 模板, Git, 测试, 监控, 自动化, 代码质量, 开发方法学)
9. 定制自己的工具箱
10. 学习 Node.js

当前端的 HTML/CSS 开发者和前端应用/JavaScript 开发者分离时, 你就快要结束这个学习过程.

关于学习的一个简短建议: 在学习抽象的技术之前, 先学习实际的底层技术. 先学 DOM, 再学 JQuery; 先学 CSS, 再学 SASS; 先学 HTML, 再学 HAML; 先学 JavaScript, 再学 coffeeScript; 先学 ES6 模板字符串, 再学 Handlebars; 先学 UI 模式, 再学 Bootstrap. 当你开始学习时, 你应该会 害怕事情隐藏的复杂性. Abstracts in the wrong hands can give the appearance of advanced skills, while all the time hiding the fact that a developer has an inferior understanding of the basics or underlying concepts.

正如我之前所建议的学习过程, 这本书的剩下部分为读者指明学习资源和工具. 这也假设你不仅要学习, 而且会将你所学到的知识和工具用于实践.一些人认为只 实践, 而其他人则建议只学习, 我建议你结合二者, 找到适合自己的方式, 但是一定要结合学习和实践! 因而不仅要阅读这本书, 而且要实践. 学习, 实践, 学习, 实践. 重复执行是因为事情变化太快, 这就是为什么学习技术的基本原理, 而不是抽象的技术是如此重要.

我在前文已经提到, 现在涌现出很多的非认证的前端编码教育/训练营, 这些成为前端开发者的途径也是由老师在课室(虚拟和实体)指导的课程, 遵循了从官方体系 (如: 教学大纲, 测验, 小测试, 项目, 团队项目, 成绩等)学习的传统风格, 我在这本书的学习指导部分提到了更多关于这方面的东西. 简单地说, Web 上有 一切你需要学习的东西(几乎没有成本), 然后, 如果你需要有人告诉你如何获取眞正免费的东西, 并且对你的学习负责, 你可以考虑一个有组织的课程. 关于其他方面, 我不知道其他任何职业可以通过互联网连接和对知识的强烈愿望来免费获取要学习的东西.

如果你要马上开始学习, 我建议你看看下面一些关于前端开发实践的概述:

- 前端指南 [read]
- 成为 Web 开发者 [read]
- Isobar 前端代码标准 [read]
- Web 基本原理 [read]
- 前端课程 [read]
- freeCodeCamp [interact]
- Planing a Front-end JS Application [watch]
- So, You Want to be a Front-end Engineer [watch]
- Front End Web Development Career Kickstart [watch][$]
- 前端 Web 开发入门 [watch][$]
- Front-End Web Development Quick Start With HTML5, CSS, and JavaScript [watch][$]
- Web 开发介绍 [watch][$]
- 前端 Web 开发基本原理 [watch][$]
- Lean Front-End Engineering [watch][$]
- A Baseline for Front-End [JS] Developers: 2015 [read]
- 了解前端 Web 开发 [watch][$]
- 前端开发精通 [watch][$]
- 没有学位的前端 Web 开发者 [watch][$]

译者补充:

- 成为一名优秀的Web前端开发者
- What makes a good front end engineer?
- How to become a web developer

第二部分: 学习

第二部分为成为一个前端开发者提供自主学习和指导学习的资源.

注意, 仅需要学习被列举出的资源, 或者一个类别的学习记录, 因为我不建议一个前端开发人员学习所有东西, 这是非常荒谬的. 选择自己行业内的专业知识, 我会尽可能让你掌握它.

译者补充:

- 前端开发笔记本
- 前端开发规范
- 适用于小团队的前端规范
- 无线 Web 开发浅谈
- 如何跟上前端开发的最新前沿
- Engineering Blogs
- 大前端工具集
- 前端收集
- Front-end Dev Bookmarks
- Github 资源收集
- github上值得关注的前端项目
- 前端资源库
- 前端人的俱乐部

自主学习

这个部分集中于个人能用来指导自己作为前端开发者的学习进度的免费和付费资源(视频训练,书籍等等).

这些资源包括免费的和付费的,付费的资源是以美元为单位结算的.

作者认为,任何有着正确的决心和奉献精神的人都能教自己如何成为一个前端开发者,除了一台能连接到Web的电脑和用于付费视频训练,书籍的现金,其它都不需要.

下面是一些我通常推荐的视频学习资料(专注技术):

- Frontend Masters
- pluralsight.com
- tutsplus.com
- lynda.com
- treehouse
- mijingo
- codeschool.com
- laracasts
- eventedmind.com
- egghead.io
- codecademy.com
- Khan Academy
- Tagtree
- Udacity [careful, quality varies]

译者补充:

- 号称最全前端开发学习资源~~学下来能上天了
- 国外有哪些高质量js技术博客?
- 你经常访问的技术社区或者技术博客（IT类）有哪些?
- 国外的前端开发社区有哪些
- 前端开发初学 书籍推荐
- MDN Web 技术文档
- MSDN Web
- SegmentFault
- 前端范
- w3cplus
- w3ctech
- w3cfuns
- w3help

- jobbole
- div.io
- v2ex
- CSDN
- cnblogs
- 博客园知识库
- 前端乱炖
- 慕课网
- 极客学院
- 网易云课堂
- The State of Web Type
- Frontend Dev Bookmarks

Internet/web

> 互联网使用网络协议套件(TCP / IP)链接全球数十亿的设备，是一个从区域到全球，由数以百万计的私人，公共，学术，商业，和政府网络组成的全球性网络系统，并通过一系列广泛的电子，无线，光纤网络技术相互链接。互联网提供了广泛的信息资源和服务，如早期的超文本文档和应用万维网(WWW)，电子邮件，电话和点对点文件共享网络。- wikipedia

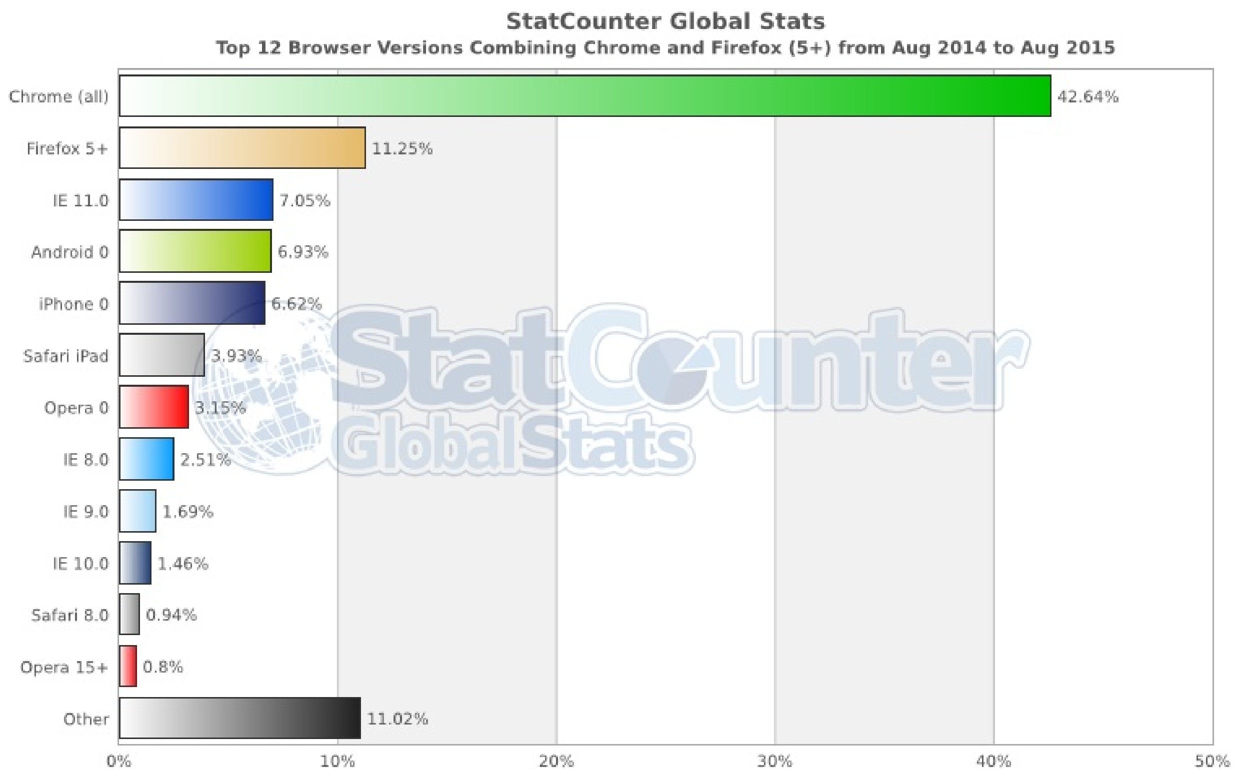
- WHAT IS THE INTERNET? or, "You Say Tomato, I Say TCP/IP" [read]
- How does the Internet work [read]
- How Does the Internet Work?" [read]
- How Does the Internet Work?" [read]
- How does the Internet Work? [read]
- How the Internet Works in 5 Minutes [watch]
- How The Web Works [watch][$]
- How the Internet Works [watch]

Web浏览器

> web 浏览器(通常被称为浏览器)是一个用于检索、展示和遍历在万维网上的信息资源的软件应用程序. 信息资源被定义成统一资源定位符(URI/URL). 它可能是网页, 图片, 视频或者一个内容片断. 超链接的出现使用户能轻松的将浏览器导航到相关的资源, 尽管浏览器主要是为了使用万维网, 但它们还可以用来访问 Web服务器在私人网络所提供的信息或文件在文件系统. - Wikipedia

主流的浏览器 如下:

1. Chrome (引擎: Blink + V8)
2. Firefox (引擎: Gecko + SpiderMonkey)
3. Internet Exploere (引擎: Trident + Chakra)
4. Safari (引擎: Webkit + SquirrelFish)



浏览器和**web**技术的演变**(API** 等**)**:

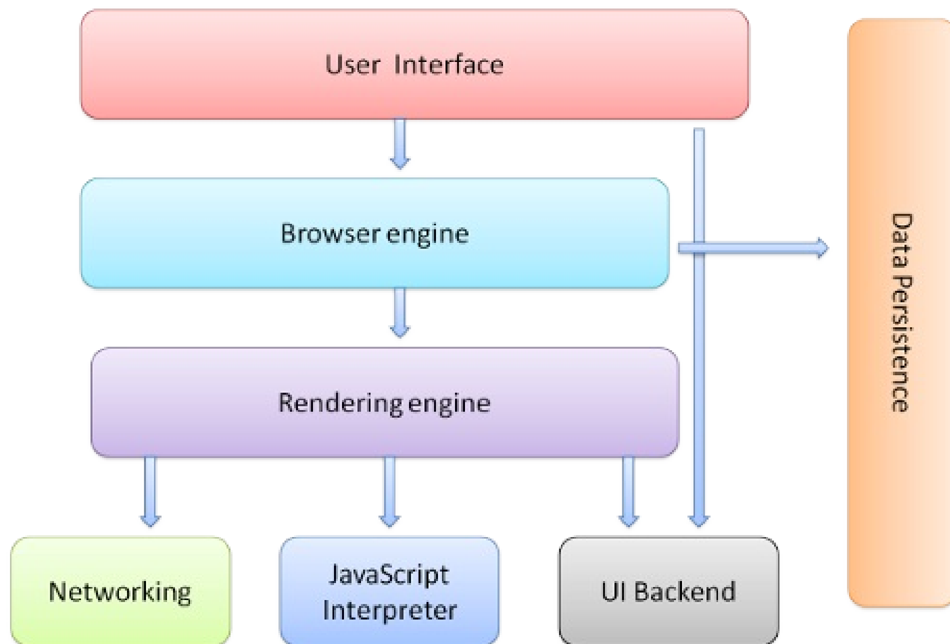
- www.evolutionoftheweb.com [read]
- Timeline of web browsers [read]

最常用的无壳浏览器:

- PhantomJS (引擎: Webkit + SquirrelFish)
- slimerjs (引擎: Gecko + SpiderMonkey)
- TrifleJS (引擎: Trident + Chakra)

浏览器怎么工作的

- 我所知道的关于浏览器和Web的20件事 [read]
- 浏览器如何工作的: 现代浏览器背后的秘密 [read]
- 快速 CSS: 浏览器是怎么组织网页的 [read]
- 浏览器是如何渲染一个网站的? [watch]



浏览器优化

- 网站性能优化 [watch]
- 浏览器渲染优化 [watch]

浏览器安全

- 浏览器安全手册 [read]
- HTML5 安全参考手册 [read]
- 前端安全 [watch]
- Web 安全: JavaScript, HTML, CSS 的使用 [read][$]
- 网络混战: 现代 Web 应用安全指南 [read]

浏览器比较

- Web 浏览器的比较 [read]

浏览器的发展

在过去, 前端开发者会花费大量的时间让代码在不同的浏览器中正常工作. 除非你必须写出兼容老版本浏览器的代码(如: IE6/IE7), 否则跟现在比起来, 这(浏览器兼容)在以前是一个很大的问题. 虽说浏览器兼容问题现在仍然存在, 但前端开发者并不用花费很多时间就能处理这类问

题. 而实际上, 现代抽离出的框架(如: JQuery, pre-processors, transpilers)已经废除了很多浏览器不一致问题.

绿色浏览器

浏览器的最新版本被认为是绿色浏览器, 也就是说, 浏览器会自动更新而不用去提示用户更新. 浏览器的自动更新摒弃了老版本浏览器进程缓慢的问题, 因为对于老版本浏览器和现代浏览器之间的共性的差异化开发是很复杂的(如: 新规范和更新速度).

浏览器选择

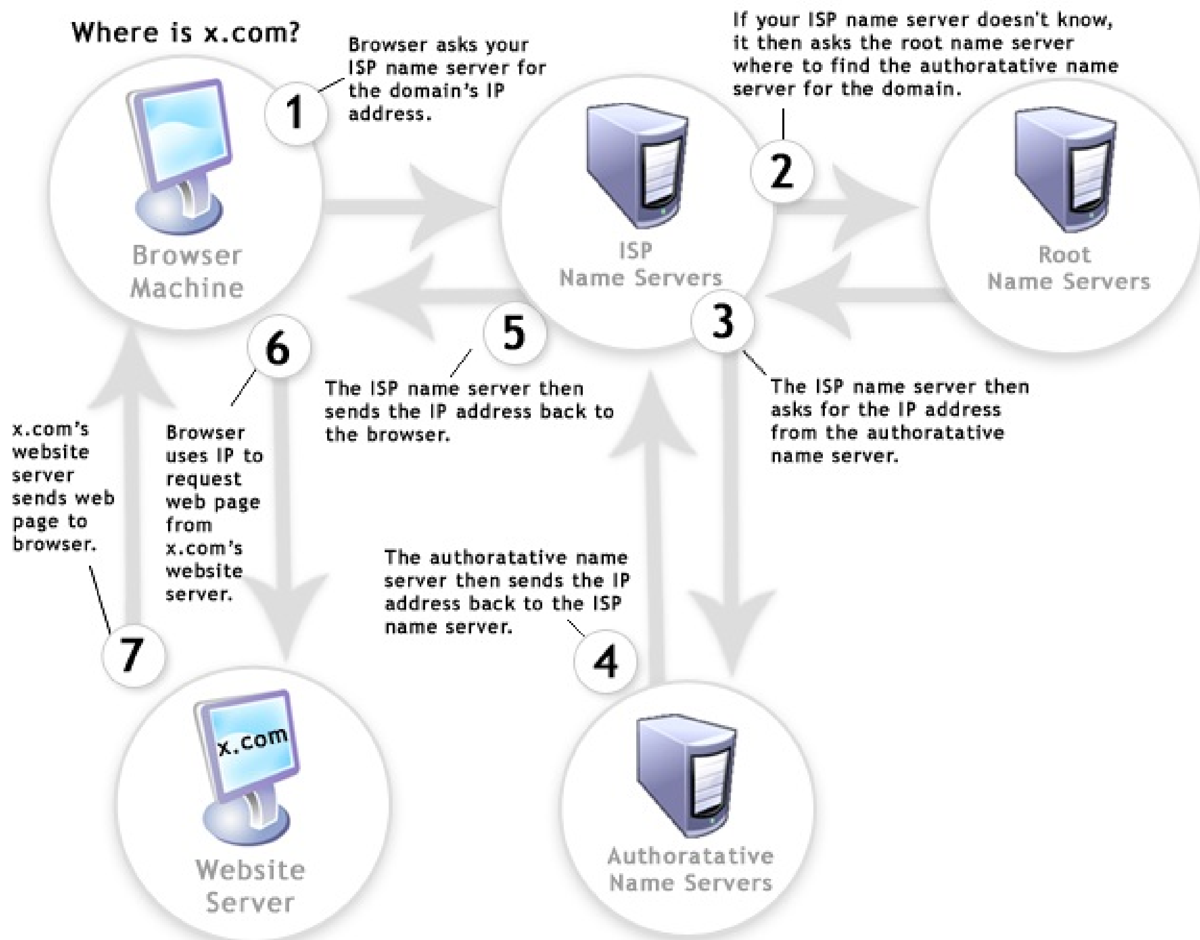
现在大多数前端开发者使用 Chrome, "Chrome 开发工具"对开发者很有用, 然而, 所有浏览器都提供了开发者工具, 所以选择一个开发用的浏览器是一个主观的问题. 更重要的问题是要了解需要支持哪些浏览器, 当你在开发的时候, 要在每个浏览器中做测试, 但无论选择哪一款浏览器都能完成开发任务, 我建议使用 Chrome 是因为 Chrome 开发工具一直在改进, 并且包含了更健全地特性.

浏览器 **Hacks**

- browserhacks.com [read]

域名系统(又称 DNS)

> 对于个人电脑、服务器或连接到互联网任何资源，或专用网络而言，域名系统(DNS)是一个分层分布式命名系统，用给每个参与的实体分配域名的方式将各种信息联系起来，更重要的是，为能访问全球的计算机服务和设备，DNS 将所需的数字 IP 地址转变为人类容易记住的域名. DNS 是大多数互联网服务的必要功能，因为这是主要的 IP 地址服务. - wikipedia



- 什么是 DNS [watch]
- DNS 是怎么工作的? [read]

HTTP/网络(包括 CORS 和 WebSockets)

> **HTTP**, The Hypertext Transfer Protocol, 即超文本传输协议, 是一个用于分布式, 协作和超媒体信息系统的应用协议, 是 WWW 数据通信的基础. - Wikipedia
>
> **CORS**, Cross-origin resource sharing, 即跨域资源共享, 是一种允许网页上受限制的资源(如: 字体)可以从该资源所在域之外的另一个域被请求. - Wikipedia
>
> **WebSockets**, WebSocket 是一种协议, 提供了在一个 TCP 连接上进行全双工通信的渠道. 在 2011 年, IETF 将 WebSocket 协议标准化为 RFC 6455, 并且 W3C 正在标准化 Web IDL 的WebSocket API. - Wikipedia

HTTP

- HTTP: 每个 Web 开发者必须知道的协议(一) [read]
- HTTP: 每个 Web 开发者必须知道的协议(二) [read]
- HTTP 基本原理 [watch][$]
- HTTP 浅谈 [read]
- 高性能网络浏览: 每个 Web 开发人员都应该了解网络和网络性能 [read]

CORS

- 60秒内的 HTTP 状态码变化 [watch]
- HTTP 访问控制 (CORS) [read]
- CORS 实战 [read][$]

WebSockets

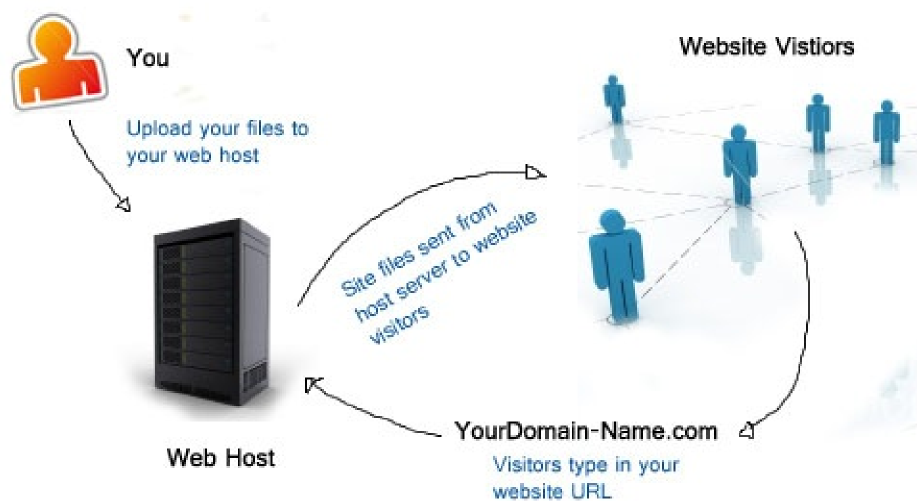
- WebSocket: 轻量级的 C/S 通信 [read][$]
- WebSocket 协议 [read]
- 用 WebSockets 连接 Web [watch]

译者补充:

- HTTP/2 资料汇总
- HTTP 权威指南 [read][RMB]
- 图解 HTTP [read][RMB]
- 图解 TCP/IP [read][RMB]

Web 主机

> Web 主机是一种网络托管服务，允许万维网访问个人和组织他们的网站，由拥有服务器的企业提供空间，或者租赁给客户使用，并提供网络连接。Web 主机也能提供数据中心空间和连接到互联网上位于数据中心的其他服务器，称为主机托管。- wikipedia



通用知识

- Web Hosting Beginner Guide [read]
- Web Hosting For Dummies [read][$]
- Ultimate Guide to Web Hosting [read]

前端开发综合学习

入门学习：

- 前端参考指南 [read]
- Web 开发者 [read]
- 前端代码标准 [read]
- Web 基本原理 [read]
- 前端课程 [read]
- FreeCodeCamp [interact]
- 开发前端 JS 应用 [watch]
- 前端工程师 [watch]
- Web 前端开发的工作内容 [watch][$]
- Web 前端开发入门 [watch][$]
- 用 HTML5, CSS, 和 JavaScript 快速进行 Web 前端开发 [watch][$]
- Web 开发介绍 [watch][$]
- Web 前端开发基础 [watch][$]
- 精益前端工程 [watch][$]
- 前端(JS)开发基线: 2015 [read]
- Web 前端开发入门学习 [watch][$]
- Web 前端开发进阶学习 [watch][$]
- 没有学位的 Web 前端开发者 [watch][$

前端简报，资讯和博客：

- shoptalkshow.com
- frontendfront.com
- webtoolsweekly.com
- O'Reilly Web Platform Radar
- The Web Platform Podcast
- The Web Ahead
- The Big Web Show
- Fresh Brewed Frontend
- Mobile Web Weekly
- Open Web Platform Daily Digest
- FRONT-END DEV weekly

译者补充：

- 前端工程系列文章
- 前端知识体系
- 前端资源大总结

- 前端入门
- WebPlatForm
- 前端开发规范手册
- Web 开发电子书下载(英)
- 对前端开发者有用的文档和指南: 1 2 3 4 5 6

用户界面和交互设计

> 用户界面设计: 用户界面设计(UI)或用户界面工程师是为机器或者软件做用户界面设计的, 如: 计算机, 家用器具, 移动设备和其它电子设备, 专注于最大限度地提高用户体验. 用户界面设计的目标是尽可能是使用户交互变得简单有效, 实现用户的操作目标(设计是以用户为中心的). - wikipedia
>
> 交互设计模式: 设计模式是一种记录解决常见设计问题解决方案的形式化方式. 建筑师 Christopher Alexander 在城市规划和建设体系结构中已经介绍了这种方式, 并已用于其他学科, 包括教学, 教育学和软件架构和设计. - wikipedia
>
> 用户体验设计: 用户体验设计(又称 UXD 或 UED 或 XD), 是通过提高可用性、可访问性以及在用户跟产品交互时的愉悦来提高用户体验的过程. 用户体验设计从完成传统的人机交互(HCI), 已经扩展到要处理产品或服务中能被用户感知的所有方面. - wikipedia
>
> 人机交互: 人机交互(HCI)不仅特别关注人和计算之间的界面, 也会研究设计和使用 Web 技术. 人机交互领域的研究人员都会去关注当前人类与计算机交互的方式和为人类与计算机提供新的交互方式的设计技术. - wikipedia

若你想要建立适当的用户界面, 我会建议你阅读以下关于设计方面的书籍:

- Designing Interfaces [read][$]
- About Face: The Essentials of Interaction Design [read][$]
- Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability [read][$]

译者补充:

- 简约至上:交互式设计四策略 [read][RMB]
- 交互设计沉思录 [read][RMB]

HTML & CSS

> **HTML** - 超文本标记语言，通常被称为 HTML，是被用于创建网页的标准标记语言。Web浏览器能将 HTML 文件渲染成可见的或者可听到的。HTML 随着线索提示，语义化地描述了网站的结构，使它成为一种标记语言，而不是编程语言。- wikipedia.org
>
> **CSS** - 层叠式样式表(CSS)是用于描述外观和格式化标记语言编写的文档的样式表语言。尽管经常被用来改变用 HTML 和 XHTML 编写的网页和用户界面的样式，但也可用于任何 XML 文档，包括纯 XML，SVG 和 XUL。跟 JavaScript 和 HTML 一样，CSS是被大多数网站用于为Web应用程序创建富有吸引力的网页，用户界面的一种基础技术，也为许多移动应用程序创建用户界面。- wikipedia.org

就像建造房子，你可以认为 HTML 是骨架，而 CSS 是油漆和装饰。

入门学习：

- codecademy.com HTML & CSS [interact]
- HTML/CSS 介绍：网页制作 [watch]
- HTML & CSS 编码学习 [read]
- CSS 布局 [read]
- Web 前端开发入门 [watch][$]
- HTML & CSS 设计与构建网站 [read][RMB]
- 快速使用 HTML5, CSS 3 & JavaScript 进行 Web 前端开发 [watch][$]
- 深入理解 HTML：语义，标准与样式 [read][$]
- CSS 定位 [watch][$]
- HTML 文档流 [watch][$]
- HTML5 & CSS3 介绍 [watch][$]
- CSS 中的绝对居中 [read]
- CSS 盒模型 [watch]
- HTML 表单结构 [watch]
- HTML 语义化：如何构建 Web 页面 [watch]

进阶学习：

- 从 CSS 1 到 CSS 4 的选择器参考 [read]
- CSS Diner [interact]
- 深入理解 CSS3 [watch][$]
- atozcss.com/ [watch]
- Flexbox 完全指南 [read]

参考/文档：

- htmlelement.info

- MDN CSS reference
- MDN HTML element reference
- cssvalues.com/
- CSS TRIGGERS...A GAME OF LAYOUT, PAINT, AND COMPOSITE
- HTML attributes reference

术语表:

- HTML 元素编程术语参考
- CSS 属性和选择器编程术语参考

标准/规范:

- W3C HTML5 标准: WWW 核心语言的第五个主要版本
- HTML 元素在线标准
- HTML 语法
- W3C HTML 规范
- 全局属性
- CSS 2.2 规范
- CSS 3 选择器
- W3C CSS 规范

CSS 架构设计:

- OOCSS [read]
- SMACSS [read][$]
- SMACSS [watch][$]
- Atomic Design [read]

编写规范:

- CSS 通用编写规范 [read]
- 开发灵活的, 持久的和可持续的 HTML 和 CSS 的标准 [read]
- cssguidelin.es [read]
- 谷歌的 HTML/CSS 编程风格指南

HTML/CSS 简报:

- HTML 5 Weekly
- CSS Weekly

译者补充:

- CSS 3 教程
- HTML & CSS 测试
- Pro HTML5 Programming

- Pro CSS3 Animation
- CSS Protips
- 精通CSS: 高级Web标准解决方案
- 高流量网站CSS开发技术
- 深入理解HTML5: 语义, 标准与样式
- HTML 5与CSS 3权威指南

SEO: 搜索引擎优化

> 搜索引擎优化(SEO)是在自然的搜索结果中, 影响一个网站或者网页可见性的过程. 一般来说, 越靠前出现(或在搜索结果页中权重高的网页)的网页, 和频繁出现在搜索结果列表中的网站, 就能获取更多来自搜索引擎的用户. SEO 会定位于不同类型的搜索, 包括图片搜索, 本地搜索, 视频搜索, 学术搜索, 新闻搜索和特定行业的垂直搜索引擎. - wikipedia

综合学习:

- David Booth: SEO 基本原理 [watch][$]
- Paul Wilson: SEO 基本原理 [watch][$]
- SEO: Web 设计者 [watch][$]
- 谷歌搜索引擎优化指南 [read]
- SEO 优化指导 [read]

JavaScript

> JavaScript 是一种高级的，动态的，无类型的和解释型的编程语言，它已经在 ECMAScript 语言规范中被标准化。跟 HTML 和 CSS 一样，JavaScript 是 WWW 内容生成的第三种必不可少的技术；大多数的网会使用Javascript，并且 Javascript 被所有现在Web浏览器支持。JavaScript 基于原型和函数优先的特点，使它成为多范型的语言，支持面向对象的，命令式的，和函数式编程风格。JavaScript 能提供 API 来处理文本，数组，日期和正则表达式，但不包括任何 I/O，如网络，存储或图形工具，对这些的依赖取决于宿主环境中嵌入了什么。
> - wikipedia.org

入门学习：

- https://www.codecademy.com/en/tracks/javascript [interact]
- JavaScript 高级程序设计 [read][RMB]
- JavaScript Enlightenment [read]
- JavaScript面向对象精要 [read][RMB]
- Speaking JavaScript [read]
- 你不知道的 JavaScript [read]
- 理解 ECMAScript 6 [read]
- JavaScript 模式 [read][RMB]
- JS.Next: ES6 [watch][$]
- Crockford on JavaScript - Volume 1: The Early Years [watch]
- Crockford on JavaScript - Chapter 2: And Then There Was JavaScript [watch]
- Crockford on JavaScript - Act III: Function the Ultimate [watch]
- Crockford on JavaScript - Episode IV: The Metamorphosis of Ajax [watch]
- Crockford on JavaScript - Part 5: The End of All Things [watch]
- Crockford on JavaScript - Scene 6: Loopage [watch]
- JavaScript 模块 [read]

进阶学习：

- JavaScript 函数式编程 [read][RMB]
- ECMA-262 by Dmitry Soshnikov [read]
- JavaScript 进阶 [watch][$]
- JavaScript 语言精粹 [read][RMB]
- 你不知道的 JS: 作用域 & 闭包 [read]
- 你不知道的 JS: this & 原型 [read]
- 你不知道的 JS: 数据类型 & 语法 [read]
- 你不知道的 JS: 异步 & 性能 [read]
- 你不知道的 JS: ES6 & Beyond [read]
- Eloquent JavaScript [read]

- 测试驱动的 JavaScript 开发 [read][RMB]
- JavaScript Allonge [read][$]
- JavaScript With Promises [read][$]
- 高性能 JavaScript [read][$]
- JavaScript 正则表达式入门 [read]
- 正则表达式使用 [watch][$]

参考/文档:

- MDN JavaScript reference

术语表/百科全书:

- JavaScript 术语表
- JavaScript 百科全书

标准/规范:

- ECMAScript® 2015 语言规范
- ECMA262 的状态, 进度和文档

编程规范:

- Node.js 风格指南
- JavaScript 开发原则
- Airbnb JavaScript 风格指南

JavaScript 简报, 新闻和播客:

- JavaScript Jabber
- JavaScript Weekly
- Echo JS
- JavaScript Kicks
- javascript.com
- FiveJS
- JavaScript Live

译者补充:

- 重新认识 JavaScript
- JavaScript 标准参考教程
- JavaScript 秘密花园
- 浅谈JavaScript系列
- 深入理解JavaScript系列
- JavaScript Promise 迷你书
- 学用 JavaScript 设计模式

- ES 6 入门(中文)
- ES 6 入门(英文)
- ES 6 In Depth 系列
- ES6 Overview in 350 Bullet Points
- JS The Right Way
- JavaScript 新手教程
- JavaScript 进阶教程
- 国外优秀 JavaScript 资源推荐
- awesome-javascript1 awesome-javascript2
- JavaScript 免费编程电子书索引(中文)
- JavaScript：40+基本的 Web 开发工具
- JavaScript Puzzlers
- JavaScript Test
- Awesome JavaScript-CN

Web 动画

综合学习:

- Web 动画的历史 [watch][$]
- Snap.svg 动画 [watch][$]
- CSS3 和 HTML5 动画 [watch][$]
- 真实世界中的 CSS 动画 [watch][$]
- HTML5+JavaScript 动画基础 [read][RMB]
- Web Animation using JavaScript: Develop & Design (Develop and Design) [read][$]
- 2014: 动画发展现状 [read]
- 学会用 CSS 创建动画 [read & watch]
- 学会用 JavaScript 创建动画 [read & watch]

标准/规范

- Web 动画

译者补充:

- Pro CSS3 Animation

DOM, BOM & JQuery

> **DOM** - 文档对象模型(DOM)用于代表和对象交互的HTML, XHTML 和 XML 文档, 是一种跨平台和语言无关性的约定. 每一份文档的所有节点被组织成一种树结构, 称为 DOM 树. DOM 对象通过使用对象上的方法被处理和操作, 一个 DOM 对象的公共接口被指定为它的应用程序编程接口(API). - wikipedia.org
>
> **BOM** - 浏览器对象模型(BOM)是一种浏览器规范, 代指 Web 浏览器暴露出的对象. 与文档对象模型不同的是, 目前并没有关于浏览器对象模型的标准和严格定义, 因而浏览器厂商可以按照他们的意愿, 采取任何方式来实现 BOM. - wikipedia.org
>
> **JQuery** - JQuery 是一个跨平台的 JavaScript 库, 其设计目的是简化客户端的 HTML 脚本操作. JQuery 也是目前最流行的 JavaScript 库, 在目前排名前1000万的网站中, 65% 的网站安装了 JQuery. JQuery 是免费的, 基于 MIT 协议的开源软件. - wikipedia.org

最理想但又难度最大的学习路径就是首先学习 JavaScript, 然后学习 DOM, 之后再学习 JQuery. 然后, 完全可以按照你所想的学习路径来安排学习的顺序. 大多数的前端开发者在了解 JavaScript 和 DOM 之后, 才会接触 JQuery. 无论采取什么学习路径, 都要确保 JavaScript, DOM 或者 JQuery 不要成为"黑盒子".

入门学习:

- DOM [read]
- codecademy.com jQuery [watch]
- jQuery 启蒙 [read]
- JQuery 入门 [watch][$]

进阶学习:

- DOM 启蒙 [read][RMB] 或 免费在线阅读
- JS & DOM 进阶 [watch][$]
- DOM 进阶: 动态网页设计技术 [read][$]
- 常见 JQuery bugs 修复 [watch][$]
- JQuery 技巧 [watch][$]
- jQuery-free JavaScript [watch][$]
- Douglas Crockford: DOM 理论--不可忽视的 API [watch]

参考/文档:

- MDN: 文档对象模型
- MDN: 事件参考
- JQuery 文档
- MDN: 浏览器对象模型
- MSDN: 文档对象模型

标准和规范:

- W3C DOM4
- DOM 在线标准
- DOM 3 事件规范
- DOM 技术报告

译者补充:

- You Might Not Need JQuery(英) 中文版

Web 字体 & 图标

> Web 字体是指在 WWW 上使用的字体. 当 HTML 文档第一次被创建时, 字体风格和样式会被每个 Web 浏览器的设置所控制, 因为直到 1995 年网景介绍了 `<font>` 标记之前, 个人网页没有控制字体显示的方式, 而 `<font>` 标记在 HTML 3.2 规范中被标准化. 然而, 被标记指定的字体必须安装在用户的电脑上, 或者是一种可以依赖的字体, 如: 浏览器默认的 sans-serif 字体或 monospace 字体. 在1996 年 发布的 CSS 1.0 规范也提供指定使用字体的功能.
>
> 1998 年, CSS 2.0 规范发布, 意图通过字体匹配, 合成和下载技术, 改善字体的选择过程, 但这些技术并没有得到使用, 并在 CSS2.1 规范中被移除了. 然而, 在 1997 年发布的 IE 4.0 增加了对字体下载的支持, 随后, CSS 3 的字体模块变包含了字体下载, 并且 Safari 3.1, Opera 10 和 Mozilla Firefox 3.5 实现了这一功能, 随后便增加了 Web 字体和所使用字体的下载. - wikipedia

综合学习:

- Web 字体排版 [watch][$]
- @font-face 快速学习指南 [read]
- 响应式排版 [watch][$]
- 用最好的字体来美化网页 [read]

无障碍设计

> 无障碍设计是指产品，设备，服务，或者环境是为残疾人设计的。无障碍设计的概念意味着与一个人的辅助技术(例如，电脑屏幕阅读器)相兼容，确保直接访问(即独立)和"间接访问"。
>
> 无障碍设计可以理解为 "能够访问"，并对一个系统或实体是有利的，其侧重于使身体残障，或有特殊需要，或要依赖辅助技术的人群能够访问 Web。然后，研究和开发无障碍设计对每个人都带来了好处。
>
> 无障碍设计不应该和可用性混淆。大多数情况下，可用性是指产品(如：设备，服务，或者环境)能在特定的环境下被特定的用户使用，来高效地实现制定目标。
>
> 无障碍设计和通用性设计是息息相关的。通用型设计是指产品的创造过程中，产品对人们是可用的，并尽可能最大范围覆盖各能力范围内的人群和各种情形下的操作，即对所有人是可访问的(无论他们访问 Web 是否有障碍)。- wikipedia

综合学习：

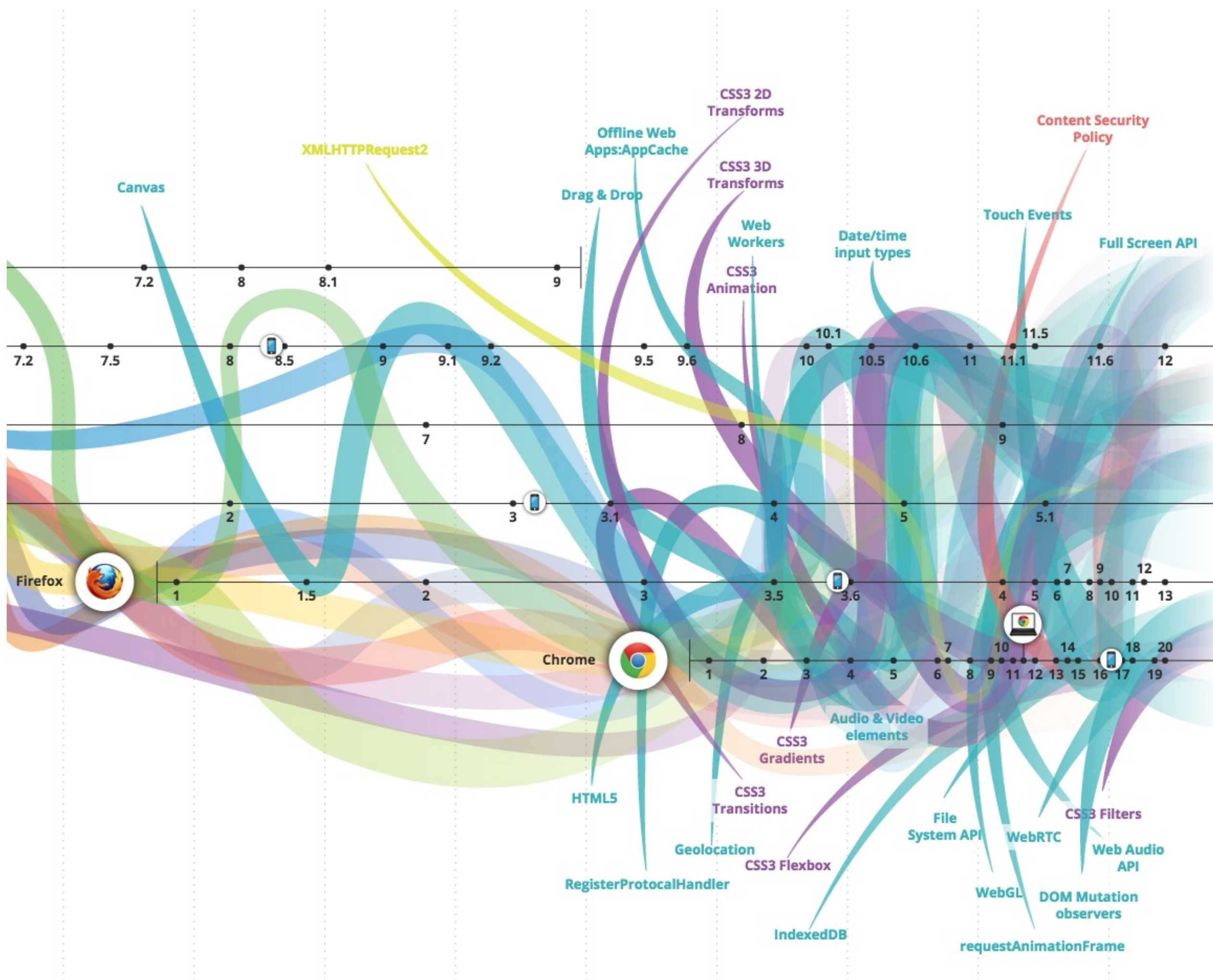
- Web应用程序通用设计 [read][RMB]
- Web 无障碍设计入门 [watch][$]
- Web 无障碍设计介绍 [read]
- 兼济天下·用户无障碍体验的可访问性设计 [read][RMB]
- UX 基础：无障碍设计 [watch][$]
- Web 无障碍设计介绍 [watch]

标准/规范：

- 无障碍设计网络倡议 (WAI)
- 网页内容无障碍设计指南 (WCAG) 的目前状态
- 富 Internet 应用程序的无障碍设计 (WAI-ARIA) 的目前状态

Web/浏览器 API

BOM 和 DOM 并不是唯一的浏览器 API, 在浏览器内部的 Web 平台上, 它们是可用的. DOM 和 BOM 并不是一切, 但是一个用于浏览器编程的接口可以被认识一个 Web 或者 浏览器 API(悲剧的是, 这些 API 曾被称为 HTML 5 API, 这会和 HTML 5 自身的规范/标准混淆, 因为 HTML 5 规范特指 HTML 5 标记语言). Web 或浏览器 API 也会包括访问设备的 API(如: `Navigator.getBattery()`), 通过平板和手机设备上的浏览器可以利用这些 API.



在适当情况下, 你应该了解和学习 Web/浏览器 API. 为了熟悉这些 API, 应该去研究HTML5test.com 对与 5 款常用浏览器对 API 新特性的支持结果, 这是一个很不错的方式.

学习文档:

- DIVE INTO HTML5 [read]
- HTML5 专业开发 [read]
- HTML5 Canvas [read]

学习视频:

- Web Audio 的乐趣 [watch]

- 用 Web Audio 给网站添加声音 [watch]

其它学习资源:

- Web Audio API [read]

此外, MDN 有很多关于 web/browser API 的信息:

- MDN Web API 参考
- MDN WebAPI - 设备访问 API 和其它有用的 API
- MDN Web APIs 接口参考

时刻记住, 并不是每一个 API 都在 W3C 或 WHATWG 的规范之内.

除了 MDN, 还列出了一些有用的资源:

- HTMl5-overview
- platform.html5.org
- HTML5 和 JavaScript API 索引

JSON(JavaScript Object Notation)

> JSON, 有时也称 JavaScript 对象表示, 是一种使用人类可读的文本传输由键值对组成的数据对象的开放格式. 对于异步浏览器/服务器通信(AJAJ), JSON 是主要的数据格式, 很大程度上代替了 XML(AJAX).
>
> 尽管最初是从 JavaScript 脚本语言衍生而来, 但是 JSON 是语言无关性的数据格式, 在许多编程语言中, 代码解析和生成 JSON 是很容易的.
>
> JSON 的格式最初是由 Douglas Crockford 指定的, 但目前却被描述成两种标准: RFC 7159 和 ECMA-404. ECMA 标准只允许被合法的语法语句描述, 而 RFC 则提供了一些语义化描述和安全考虑. JSON 的官方网络媒体类型 application / JSON, 扩展名是 .json. - wikipedia.org

综合学习:

- json.com [read]
- 什么是 JSON [watch]
- JSON 权威指南 [read][$]

参考/文档

- json.org/ [read]

标准/规范:

- ECMA-404 JSON 数据交互格式
- RFC 7159 JSON 数据交互格式

架构设计:

- JSON API

译者补充:

- JSON 系列文章

静态网页生成器

静态网页生成器, 是使用服务器端代码编写(如: ruby, php, python, nodeJS 等...), 用静态文本数据 + 模板, 生成从服务器发送到客户端的静态 HTML 文件.

综合学习:

- 静态网页生成器 [read]

前端应用架构设计

- JavaScript Web 应用开发 [read][RMB]
- 用 React & Ampersand 构建 APP [watch][$]
- Human JavaScript [read]
- JavaScript 应用程序编程 [read]
- 构建现代单页应用 [watch][$]
- JavaScript 函数式编程 [watch][$]
- JavaScript: 模块 [read]
- Web UI 架构设计 [watch][$]
- Web 应用设计指南 [read]
- UI 架构设计 [watch][$]
- Terrific [read]
- Nicholas Zakas: 可扩展的 JavaScript 应用架构 [watch]
- 大型 JavaScript 应用架构设计模式 [read]
- 静态应用指南 [read]

接口/API 设计

数据(类似 JSON) API:

- Build APIs You Won't Hate [$][read]
- JSON API [read]

JavaScript API

- Writing JavaScript APIs [read]
- Designing Better JavaScript APIs [read]

译者补充

- HTTP API 设计指南
- 用 JSON 构建 API 的标准指南

Web 开发者工具

> Web 开发者工具允许开发者测试和调试代码，它们不同于网站生成器和 IDE，因为 Web 开发者工具不直接参与网页的创建，而是用于测试网站或 Web 应用的用户界面接口的工具.
>
> Web 开发者工具是浏览器的加载项或内置功能. 当今最流行的web浏览器，Google Chrome, Firefox, Opera, Internet Explorer, 和 Safari 都内置工具用于帮助开发者，并且在各自的插件下载中心，还提供很多额外的加载项.
>
> Web 开发者工具允许开发者使用各种各样的 Web 技术，包括 HTML, CSS, DOM, JavaScript 和 其它浏览器能够处理的组件. 由于日益增长的需求，更多流行的 Web 浏览器包括了更多面向开发人员的功能.

尽管大多数浏览器都配备了开发者工具，但是 谷歌开发者工具 是目前谈论最多，应用最广泛的开发者工具.

我建议学习和使用 谷歌开发者工具，因为谷歌开发者工具周围有学习 Web 开发者工具最好的资源.

学习使用谷歌开发者工具:

- Explore and Master Chrome DevTools
- Chrome 开发者工具 [watch][$]
- 使用 Chrome 开发者工具 [watch][$]

谷歌开发者工具文档:

- Per-Panel 文档
- 命令行接口参考
- 键盘 & UI 快捷键参考
- 设置

新闻/简报/博客/建议:

- Dev Tips

译者补充:

- Chrome Devtools Tips & Tricks
- Chrome开发者工具不完全指南
- Chrome 开发工具指南

命令行

> 命令行接口或命令语言解译器(CLI), 也称命令行用户界面, 控制台用户界面和字符用户界面(CUI), 是一种用户以连续的文本(命令行)的形式向程序提出需求, 与计算机程序交互的方式. - wikipedia

入门学习:

- codecademy: Learn the Command Line [watch]
- The Command Line Crash Course [read]
- Meet the Command Line [read][$]
- Learn Enough Command Line to Be Dangerous [read][free tp $]
- Command Line Power User [watch]

进阶学习:

- Advanced Command Line Techniques [watch][$]

Node.js

> Node.js 用于开发服务端 Web 应用，是一个开源的，跨平台的运行时环境. Node.js 应用由 JavaScript 编写，可以在 OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM AIX, IBM System z and IBM i 上的 Node.js 运行时环境运行. Node.js 的开发和维护有 Node.js 基金会提供支持，这是 Linux 基金会的一个合作项目.
>
> Node.js 提供一个事件驱动的体系架构和非阻塞的 I/O 设计来优化应用程序的吞吐量和实时web应用程序的可伸缩性，它使用谷歌的 V8 JavaScript 引擎来执行代码，并有大量的由 JavaScript 编写的基础模块. Node.js 包含内置的模块，允许应用程序作为一个web服务器而不依赖类似 Apache HTTP Server, Nginx 或 IIS 的软件. - wikipedia

入门学习：

- 从事件角度介绍 Node.js [watch][$]
- Node 的艺术 [read]
- Node.js 基础 [watch][$]
- io.js 和 Node.js 入门 [watch][$]
- 全面学习 Nodes [watch]
- Node 初学者书籍 [read][$]
- Node.js 介绍 [watch][$]
- Node.js 实战 [read][$]
- Nodeschool.io [code]

译者补充：

- Node.js 经典入门教程(中文版)
- 从零开始学 Node.js 系列文章
- Node.js 入门
- Node.js 中文文档：1 2 3
- Node.js 风格指南
- Express 中文文档：1 2
- Express 指南
- Koa中文文档：1 2 3 4
- NodeCloud
- Node面试题
- Node.js 中文资料导航
- Node.js 开发常用资源(awesome-nodejs)1 开发资源2 开发资源3
- Node-books

React

> **React** - 用于构建用户界面的JAVASCRIPT库
>
> 仅仅是**UI** - 许多人使用React作为MVC架构的V层。 尽管React并没有假设过你的其余技术栈， 但它仍可以作为一个小特征轻易地在已有项目中使用
>
> 虚拟**DOM** - React为了更高超的性能而使用虚拟DOM作为其不同的实现。 它同时也可以由服务端Node.js渲染 — 而不需要过重的浏览器DOM支持
>
> 数据流 - React实现了单向响应的数据流，从而减少了重复代码，这也是它为什么比传统数据绑定更简单。

入门教程:

- React 入门实例教程
- React 介绍及实践教程 此文中的demo实现: demo
- React 入门教程
- Learning React
- SurviveJS - Table of Contents

进阶学习:

- 深入浅出 React 系列
- React.js 生态系统概览
- 详解React Flux架构工作方式
- Awesome React
- AlloyTeam: React 专栏

手册参考:

- React 速查手册
- React Cheat Sheet
- React Style Guide

工具:

- Awesome Redux
- Webpack 中文文档
- webpack学习笔记
- Webpack傻瓜式指南
- React Webpack Cookbook
- Webpack
- React Toolbox

React Native:

- React Native Lession
- Awesome React Native
- React Native 探索系列
- React Native专题
- React Native 学习指南
- React Native 中文版

模块加载和依赖管理

综合学习：

- 用 Browserify 创建 JavaScript 模块 [watch][$]
- Webpack 基本原理 [watch]
- browserify-handbook [read]
- ES6 模块 [read]

参考/文档：

- browserify
- system.js
- webpack

译者补充：

- React Webpack cookbook
- 详解前端模块化工具-Webpack

包管理器

> 包管理器或包管理系统是一系列软件工具的集合，这些软件工具用和电脑操作系统一致的方式，使应用的安装，升级，配置和删除软件包的过程自动化，它通常维护一个数据库软件的依赖和版本信息，防止软件不匹配和无法跟踪。- wikipedia

综合学习：

- Bower 基本原理 [watch][$]
- 包管理器：前端开发人员入门指南 [read]
- NPM: 包的上传和运行 [watch][$]
- NPM 基础 [watch][$]
- NPM 书籍 [read]
- NPM & Bower: 依赖版本管理 [read]

参考/文档：

- Bower
- jspm.io
- NPM

版本控制

> 软件配置管理, 版本控制的一个组成部分, 也称为校正控制或源码控制, 用于管理文档, 计算机程序, 大型网站和其它信息集合的变化. 变化通常被定义为一串数字或字母代码, 被称为 "版本编号", "版本标识", 或简称"版本". 举个例子, 初始的文件集合是"版本1", 当第一个改变文件时, 就变成了"版本2"等等. 每一个版本都和一个时间戳和做出改变的人联系在一起. 版本可以被比较, 恢复和跟其它文件合并. - wikipedia

入门学习:

- codeschool.com [interact]
- Git 基本原理 [watch][$]
- 正确使用 Git [read]
- Git 介绍 [read]
- Git 权威指南 [read]

进阶学习:

- Git 进阶 [read]
- Git 权威指南 [read]

参考/文档:

- https://git-scm.com/doc

译者补充:

- Pro Git 教程
- Git 教程
- git-scm(英) 中文版
- my-git

构建和任务自动化

> 构建自动化是软件构建和相关流程的自动化过程，包括：将计算机源码编译成二进制代码，打包二进制代码和运行自动化测试。- wikipedia

综合学习：

- 用 Gulp.js 进行 JavaScript 自动构建：[watch][$]
- Gulp 入门 [read][$]
- Gulp 快速入门 [watch][$]
- 学习 Gulp - 前端工厂入门 [read]
- Gulp 基础 [watch][$]
- 使用 npm 作为任务运行器 [watch][$]

参考/文档：

- gulp

Gulp 是非常棒的构建工具，然后，你可能仅仅只需要 `npm run`。在你的应用程序栈变得负责之前，问问你自己，`npm run` 是否能完成自动化构建。如果你需要更多，可以同时使用 `npm run` 和 Gulp。

推荐阅读：

- NPM 使用指南
- NPM 的任务自动化管理
- 构建工具 VS NPM 脚本：为何不使用两者
- 使用 NPM 作为下一个项目的构建系统
- 怎么将 NPM 作为构建工具使用

译者补充：

- Gulp不完全入门教程

网站性能优化

> Web 性能优化, WPO, 或网站优化是提高用户浏览器的网站加载和显示速度的知识领域. 由于网速整体提高了, 很适合网站的管理者和维护者去考虑网站呈现给访问者所花费的时间了. - wikipedia

综合学习:

- 网站性能 [watch][$]
- 高性能网站建设指南：前端工程师技能精髓 [read][RMB]
- 高性能网站建设进阶指南：Web开发者性能优化最佳实践 [read][RMB]
- Web 性能实践日志 [read][RMB]
- JavaScript 性能宝石 [read]
- 性能日历 [read]
- 页面速度见解规则 [read]
- 网站性能优化 [watch]
- 浏览器渲染优化 [watch]
- Using WebPageTest [read][$]
- Web 性能权威指南 [read][RMB]
- perf.rocks

译者补充:

- 谷歌的性能优化指南
- 前端性能优化指南
- AlloyTeam: Web 性能优化专栏

JS 测试

> 单元测试 - 在计算机程序中，单元测试是一种软件测试方法，通过独立的代码单元，一个或多个计算机程序模块的集合，和相关联的控制数据，使用程序和操作过程进行测试，以确定它们是否适合使用. 直观地说，可以将一个单元视为应用程序最小的，可测试的一部分. - Wikipedia
>
> 功能测试 - 功能测试是一个质量保证(QA)的过程，也是一种基于在软件组件测试规范之下的测试案例的黑盒子测试. 功能测试会检查程序的输入和输出，但很少去考虑内部地程序结构(跟白盒子测试不同). 功能测试通常描述了系统"做什么". - Wikipedia
>
> 集成测试 - 在软件测试中，集成测试(也称集成和测试，缩写为 I&T)是各个软件模块结合起来，作为一个整体进行测试的阶段，之后便是单元测试和验证测试. 集成测试把已经通过的单元测试作为输入模块，将它们组织成更大的聚集，在集成测试中应用计划用于这些聚集而被定义的测试，并提供为集成测试做好准备的集成系统作为其输出. - Wikipedia

综合学习：

- 测试驱动的 JavaScript 开发 [read][RMB]
- 编写可测试的 JavaScript 代码 [read][RMB]
- JavaScript Testing Recipes [read][$]
- 前端优先: JavaScript 的测试和原型设计 [watch][$]
- 测试驱动的 JavaScript [watch][$]

无壳浏览器

> 无壳浏览器是指没有图形用户界面的 Web 浏览器.
>
> 无壳浏览器拥有一个和受欢迎的 Web 浏览器相似的环境, 并提供了网页的自动化控制, 但要通过命令行接口或使用网络通信工具执行. 对于测试网页, 无壳浏览器是非常有用的, 因为和普通浏览器一样, 它们能渲染和理解 HTML, 包括样式元素, 如: 页面布局, 颜色, 字体选择, JavaScript 的执行和 AJAX, 但是当使用其它方法时, Ajax 通常就不可用了. 在 2009 年, 谷歌声称使用无壳浏览器有助于搜索引擎使用 Ajax 从其它网站索引内容. - wikipedia

- PhantomJS Cookbook [read][$]
- Getting Started with PhantomJS [read][$]
- Rapid PhantomJS [watch][$]
- PhantomJS for Web Automation [watch]

离线开发

离线开发(又称离线优先)是一个领域常识和围绕设备并不总是连接到互联网或电源的开发实践的讨论.

综合学习:

- offlinefirst.org [read]
- HTML5 离线 Web 应用 [read]
- 离线优先 [read]
- 创建离线应用你需要知道的一切 [read]

安全

- 浏览器安全手册 [read]
- HTML5 安全手册 [read]
- 前端安全 [watch]
- Security for Web Developers: Using JavaScript, HTML, and CSS [read][$]
- 现代 Web 应用安全指南 [read][$]
- Web 安全基础 [read]

多平台开发



一个网站或应用不仅能运行在各种台式机，笔记本电脑，平板和手机，还能运行于少部分其它设备(手表，温控器，电冰箱等等). 你将怎么决定支持哪些平台和为支持这些平台，怎么去开发，这被称为多平台开发策略. 接下来，我会列出常见的多平台开发策略:

- 创建 响应式 Web 设计 (RWD) 网站/APP
- 创建 RESS (基于服务端组件的响应式 Web 设计) 网站/APP
- 创建 自适应/渐进增强地 网站/APP
- 对每一个或每一组平台建立一套网站，Web 应用，本地应用或混合应用

- 尝试修改你用策略1, 策略2或策略创建的应用. 这可能和点缀与屏幕大小无关的部分 UI 一样简单, 也可以试图完全支持其他平台与整个 UI.

入门学习:

- 自适应网页设计 [read][$]
- 设计与渐进增强 [read]
- 响应式排版 [watch][$]
- 响应式 Web 设计 [watch][$]
- 响应式 HTML 邮件设计 [watch][$]
- Designing Multi-Device Experiences: An Ecosystem Approach to User Experiences across Devices [read][$]
- 响应式 Web 设计原理 [watch]
- 响应式图片 [watch]
- 移动 Web 开发 [watch]

译者补充:

- 移动 Web 调试工具: weinre
- Responsive demos & tutorials
- 打造最舒适的 Webview 调试环境

Directed learning

This section focuses on directed learning via schools, courses, programs and bootcamps.

Directed learning

The table below contains instructor led, paid, front-end courses, programs, schools, and bootcamps.

If you can't afford a directed education, a self directed education using screencasts, books, and articles is a viable alternative to learn front-end development for the self-driven individual.

company	course	price	on site	remote
Iron Yard	Front End Engineering	12,000	multiple locations	
Udacity	Front-End Web Developer Nanodegree	200 monthly	multiple locations	yes
The New York Code + Design Academy	Front End 101	2,000	New York, NY	
Portland Code School	Advanced Front-end Developer Tools	2,000	Portland, OR	
BLOC	Become a Frontend Developer	4,999		yes
Thinkful	Frontend Web Development	300 per month		yes
General Assembly	Frontend Web Development	3,500	multiple locations	
Hackbright Academy	Front-End Web Development	3,000	San Francisco, CA	
HackerYou	Front-end Web Development Immersive	7,000 - 7,910	Toronto, Canada	
The Flatiron School	Introduction to Front-End Web Development	3,500	New York, NY	
Austin Coding Academy	The Front End Track	1,490 per class	Austin, TX	
Codeup	Night Front-End Bootcamp	8,500	San Antonio, Texas	
Betamore	Front-end Web Development	8,500	Baltimore, MD	
Codify Academy	Front-end Web Development	4,000	multiple locations	
DecodeMTL	Learn Front-end Web Development	2,500	Montreal, QC	
gr8code	Front-End Bootcamp	10,000	Tampa, FL	
LearningFuze	Part-Time Front-End Development Course	2,500	Irvine, CA	

前端开发者从哪里学

关于前端开发, 独立学习会慢慢地变得没有意义. 前端开发的高级从业人员已经产出足够多的内容, 而你只要通过关注前端"资讯"(简报, 资讯 & 博客), 跟着社区学习就行.

前端简报，资讯网站 & 博客

综合的前端简报，资讯 & 博客

- shoptalkshow.com
- frontendfront.com
- webtoolsweekly.com
- O'Reilly Web Platform Radar
- The Web Platform Podcast
- The Web Platform Podcast
- The Web Ahead
- The Big Web Show
- Fresh Brewed Frontend
- Mobile Web Weekly
- Open Web Platform Daily Digest
- FRONT-END DEV weekly
- Web Design Weekly
- Front-end News in 5 Minutes
- TTL
- Viewsources
- Web Components Weekly

HTML/CSS 简报：

- HTML 5 Weekly
- CSS Weekly

JavaScript 简报，资讯 & 博客：

- Javascript Jabber
- JavaScript Weekly
- Echo JS
- JavaScript Kicks
- javascript.com
- FiveJS
- JavaScript Live

图形和动画简报 & 博客：

- Motion and Meaning
- SVG Immersion Podcast
- Web Animation Weekly
- Responsive Images Community Group Newsletter *

译者补充:

- 奇舞团
- FEX
- 开发者头条
- 码农周刊
- OurJS
- 编程狂人
- 一周拾遗
- 设计匠艺

第三部分

第三部分会简单地讨论一些前端开发工具的使用.

为确保你理解一套工具所属的类别, 建议在此之前先研究工具本身.

注意, 仅仅是一个工具列表, 或一个类别的工具记录, 但这并不等于我断言, 前端开发人员应该学习它并使用它. 选择自己的工具箱, 我只是提供常见的工具箱选项.

译者补充:

- Web 前端开发资源汇总

常用前端开发工具

开发工具:

- screensiz.es
- placehold.it
- codeKit
- prepros
- Browsersync
- ish. 2.0.
- Wraith

在线代码编辑:

- jsbin.com
- jsfiddle.net
- liveweave.com
- es6fiddle.net
- codepen.io
- Plunker

查找工具:

- stackshare.io
- javascripting.com
- built with
- microjs.com
- The Tool Box
- unheap.com
- stylesheets.co

DOC/API 浏览工具

用于浏览开发者文档和 API 文档的工具.

- devdocs.io
- Dash [OS X, iOS][$]
- Velocity [Windows][$]
- Zeal [Windows, Linux]

SEO 工具

- Google Webmasters Search Console
- Varvy SEO tool
- Keyword Tool

原型和框架工具

创建原型和框架:

- Balsamiq Mockups [$]
- Justinmind [$]
- UXPin [free to $]

合作/展示:

- InVision [free to $]
- myBalsamiq [$]
- conceptboard [free to $]

图表工具

- Cacoo [free to $]
- gliffy [free to $]
- draw.io [free to $]

HTTP/网络工具

- Charles [$]
- Fiddler
- Postman
- Chrome DevTools Network Panel

了解代码编辑器

> 源代码编辑器是一个文本编辑程序，专门为编辑计算机程序源代码的程序员而设计的，它可能是一个独立的应用程序或内置在集成开发环境(IDE)或web浏览器中．源代码编辑器是最基本的编程工具，,作为程序员的基本工作就是编写和编辑源代码．- Wikipedia

前端代码可以被一个简单的文本编辑应用程序(如: Notepad 或 TextEdit)，但是，大多数前端人员使用专门为一种编程语言而设计的代码编辑器编辑．

可以这么说，代码编辑器有各种各样的类型和大小．选择一个编辑器是主观行为。选择一个，学习它的使用,然后继续学习 HTML, CSS 和 JavasCript DOM.

但是，我一直相信，编辑器应该有如下特点：

- 一份不错的关于如何使用编辑器的文档
- 报告 HTML, CSS 和 JavaScript 代码的质量
- 为 HTML, CSS 和 JavaScript 提供语法高亮
- 为 HTML, CSS 和 JavaScript 提供代码自动完成
- 通过插件的方式自定义编辑器架构
- 有大量的第三方仓库/插件社区，能够用于自定义编辑器
- 轻量，简单，不耦合代码(不需要编辑代码等等)

我个人推荐将下列的插件和 Sublime Text 一起使用：

- Package Control
- AutoFileName
- SublimeLinter
  - SublimeLinter-json
  - SublimeLinter-jslint
  - SublimeLinter-html-tidy
- SideBarEnhancements
- Terminal
- BracketHighlighter
- Color Highlighter
- CSS3
- HTMLAttributes
- StringEncode
- HTML-CSS-JS Prettify

Sublime 的学习资源：

- Sublime Productivity [read][$]
- Sublime Text Power User Book [read + watch][$]

- Sublime Text 3 From Scratch [watch][$]
- Perfect Workflow in Sublime Text 2 [watch][requires login, but free]

代码编辑器:

- Atom
- Brackets
- Sublime Text [$]
- WebStorm [$]
- Visual Studio Code

在线合作的代码编辑器:

- jsbin.com [free to $]
- jsfiddle.net
- liveweave.com
- es6fiddle.net
- codepen.io [free to $]
- Plunker

在线代码编辑器:

- codeanywhere [free to $]
- Koding [free to $]
- Clound9 [free to $]

浏览工具

JS 浏览工具:

- URI.js
- platform.js
- history.js
- html2canvas

参考工具**(**查看浏览器是否支持某特性**)**

- caniuse.com
- HTML5 Please
- HTML5 test
- Browserscope
- webbrowsercompatibility.com
- iwanttouse.com/
- Platform status
- Browser support for broken/missing images
- Big JS-Compatibility-Table
- jscc.info
- What Web Can Do Today

浏览器开发**/**调试工具**:**

- Opera Dragonfly
- Safari Web Inspector
- Firefox Developer Tools
- Chrome Developer Tools(aka DevTools)
  - Per-Panel Documentation
  - Command Line API Reference
  - Keyboard & UI Shortcuts Reference
  - Settings
- IE Developer tools(aka F12)
- vorlon.js

同步浏览工具**:**

- Browsersync

浏览器编码工具**(**判断用户的浏览器是否支持某特性**):**

- Modernizr
- ES Feature Tests

浏览器的各种 **polyfills/shims:**

- webcomponents.js
- webshim
- HTML5 Cross Browser Polyfills
- console-polyfill
- socket.io
- sockjs

浏览器承载测试/自动化:

- browserstack [$]
- browserling[$]
- Sauce labs [$]
- Selenium
- CrossBrowserTesting.com

无壳浏览器:

- PhantomJS
- slimerjd
- TrifleJS

无壳浏览器的自动化工具:

- nightwatchjs
- casperJS
- Nightmare
- gremlins.js

浏览器 **hacks:**

- browserhacks.com

HTML 工具

HTML 模板:

- HTML5 Boilerplate
- Mobile boilerplate
- Web Starter Kit Boilerplate & Tooling for Multi-Device Development
- dCodes
- HTML5 Bones
- Email-Boilerplate
- Pears

HTML polyfill:

- html5shiv

Transpiling:

- jade
- HAML
- Markdown

文档参考:

- HTML Interfaces Browser Support
- HTML Entity Lookup
- HTMLElement.info
- ELements
- Elelemt Attributes
- HTML Arrows

Linting/hinting:

- html5-lint
- HTMLHint
- html-inspector

HTML 优化:

- HTML Minifier

HTML 在线生成工具:

- tablesgenerator.com

编写规范:

- Principles of writing consistent, idiomatic HTML
- HTML code guide

Workflow:

- Emmet

本月 **HTML** 仓库在**Github**的趋势**:**

- https://github.com/trending?l=html&since=monthly

CSS 工具

桌面 & 移动应用 **CSS** 框架:

- Semantic UI
- Foundation
- Bootstrap
- Metro UI
- Pure.css
- Concise
- Materialize
- Material Design Lite(MDL)
- Base

移动应用 **CSS** 框架:

- Ratchet

CSS 重置:

> CSS 重置(或重置 CSS)是一个很小的, 被压缩的 CSS 规则集合, 用于重置所有 HTML 元素的样式. - http://cssreset.com/

- Eric Meyer's "Reset CSS" 2.0
- Normalize

Transpiling:

- SASS/SCSS
- stylus
- PostCSS & cssnext
- rework & myth
- pleeease.io

参考文档:

- css3test.com
- css4-selectors.com
- css3clickchart.com
- cssvalues.com
- CSS TRIGGERS
- MDN CSS reference
- overapi.com CSS cheatsheet

Linting/hinting:

- CSS Lint
- stylelint

代码格式化/美化:

- CSScomb
- cssfmt

优化:

- csso
- clear-css
- cssnano

CSS 在线生成工具:

- Ultimate CSS Gradient Generator
- Enjoy CSS
- CSS matic
- patternify.com
- patternizer.com
- CSS arrow please
- flexplorer
- Flexbox Playground

CSS 架构:

- oocss [read]
- SMACSS [read][$]
- Atomic Design [read]

编写规范:

- idiomatic-css [read]
- CSS code guide [read]
- cssguidelin.es [read]
- Google HTML/CSS Style Guide

本月 **CSS** 仓库在**Github**的趋势:

- https://github.com/trending?l=css&since=monthly

DOM 工具

DOM 库/框架:

- jQuery
- Zepto.js
- keypress
- clipboard.js
- tether.io

DOM 性能分析:

- DOMMonster

参考文档:

- DOM Browser Support
- DOM Events Browser Support
- HTML Interfaces Browser Support
- Events
- MDN Document Object Model (DOM)

DOM polyfills/shims:

- dom-shims
- Pointer Events Polyfill: a unified event system for the web platform

虚拟 DOM:

- jsdom
- virtual-dom

JavaScript 工具

JS 库:

- lodash
- underscore.js
- Moment.js
- string.js
- Numeral.js
- accounting.js
- xregexp.com
- Math.js
- wait
- async
- format.js
- Chance

编译转换 (ESX to ESX):

- Babel

JavaScript 兼容性检查:

- jscc.info

代码检查:

- jshint
- eshint
- JSlint
  - jslinterrors.com

单元测试:

- Mocha
- QUnit
- Jasmine
  - Jest

单元测试的断言库:

- should.js
- Chai
- expect.js

单元测试监控，存根和模拟：

- sinon.js

编码规范检查：

- JSCS

代码格式化/美化：

- jsfmt
- esformatter
- js-beautify

性能测试：

- jsperf
- benchmark.js

可视化，静态分析，复杂性，覆盖工具：

- jscomplexity.org
- istanbul
- Blanket.js
- Coveralls [$]
- Plato
- escomplex
- Esprima

优化：

- UglifyJS 2

混淆：

- Javascript 混淆器
- JScrambler [$]

在线代码编辑器：

- jsbin.com
- jsfiddle.net
- es6fiddle.net
- plnkr.co

在线正则表达式编辑器/可视化工具：

- regex101

- regexper
- debuggex
- RegExr

编码规范:

- Node.js 规范指南
- JavaScript 编码原则
- JavaScript 规范指南

本月 **JS** 仓库在 **Github** 的趋势:

- https://github.com/trending?l=javascript&since=monthly

NPM 上被依赖最多的包:

- https://www.npmjs.com/browse/depended

静态网页生成器工具

JS 网页生成器:

- Metalsmith
- harp

JS 博客网站生成器:

- hubpress.io
- Hexo.io

网站生成器列表:

- staticsitegenerators.net
- www.staticgen.com

APP(桌面, 移动, 平板等) 工具

前端 **App** 框架:

- AngularJS & Batarang
- Backbone & Marionette
- React & React-router & Flux & React Developer Tools
- Vue.js & vue-loader & vue-router
- Ember & Ember Inspector
- Aurelia
- Polymer & Iron Elements & Paper Elements

全栈 **JS App** 平台:

- Meteor
- Hood.ie
- MEAN

移动 **Web UI/**网站**/App** 框架:

这些解决方案可以被用到任何地方, 包括 Web 视图(Web 平台和浏览器引擎等) APP.

- Ratchet
- Kendo UI Mobile
- Mobile Angular UI
- Framework7

本机混合移动 **webview** 框架:

典型解决方案是使用 Cordova, crosswalk, 或者自定义 Webview 作为本机 API 的桥梁.

- ionic
- onsen.io

本机混合移动开发 **webview** 环境**/**平台**/**工具:

典型解决方案是使用 Cordova, crosswalk, 或者自定义 Webview 作为本机 API 的桥梁.

- AppBuilder [$]
- Monaca [$]
- Adobe PhoneGap [$]
- kony [$]
- ionic hub [free to $]
- Taco
- manifoldJS

- cacoon [free to $]

本机桌面 **webview** 应用框架:

- NW.js
- Electron

本机移动应用框架 **(**又称 **JavaScript** 本机应用**)**

解决方案不使用浏览器引擎或 Webview, 而是利用 JS 引擎作为运行环境去编译 JavaScript, 并能调用本机的 API. UI 则使用本机的 UI 组件进行构造.

- NativeScript
- React Native
- tabris.js [free to $]
- trigger.io [$]

参考**:**

- todomvc.com

App seeds/starters/boilerplates:

- React Starter Kit
- Ember starter-kit
- NG^-starter
- Angular 2 Webpack Starter
- hjs-webpack

脚手架工具

脚手架是指为整个应用生成一个初始化的模板, 而不是生成访问数据库的代码.

- Yeoman
- Slush

模板工具

Just Templating:

- Mustache.js
- Handlebars
    - htmlbars
- Nunjuncks
- Transparency
- doT.js
- Hogan.js

Templating and reactive data binding:

- Rivets.js
- paperclip.js
- riot
- vue.js
- ractive.js
- react.js
- RxJS
- knockout
- jquerymy.js

Templating to Virtual DOM:

- JSX
- t7

译者补充:

- Handlebars 入门
- Handlebars 系列文章

UI 部件 & 组件工具

桌面 & 移动:

- Kendo UI [free to $]
- Webix [$]
- Semantic UI
- Metro UI
- Bootstrap
- Materialize
- Material UI
- Polymer Paper Elements

桌面 (NW.js 和 Electron):

- photonkit
- React UI Components for OS X El Capitan and Windows 10

专注移动:

- Ratchet
- Kendo UI Mobile
- Mobile Angular UI
- Framework7

数据可视化工具

JS 库:

- d3
- sigmajs

**部件 & 组件:</h5>

- Chart.js
- C3.js
- Google Charts
- chartist-jsj
- amCharts [$]
- Highcharts [Non-commercial free to $]
- FusionCharts [$]
- ZingChart [free to $]
- Epoch

服务:

- Datawrapper
- infogr.am [free to $]
- plotly [free to $]
- ChartBlocks [free to $]

图形工具

常见图形工具:

- Two.js
- Fabric.js

画布:

- Paper.js
- EaselJS

SVG:

- svg.js
- Snap.svg
- Raphaël
- d3

Webgl:

- three.js
- pixi.js

动画工具

- Velocity.js
- snabbt.js
- TweenJS
- Dynamics.js
- GreenSock-JS

Polyfills/shims:

- web-animations-js

JSON 工具

JSON 在线编辑器:

- JSONmate

JSON 查询工具:

- DefiantJS
- ObjectPath
- JOSN Mask

生成模拟 JSON 工具:

- JSON Generator
- Mockaroo

JSON API 在线模拟工具:

- Mocky
- FillText.com
- JSONPlaceholder
- mackable.io

**JSON API 本地模拟工具:</h5>

- json-server

JSON 规范/模式:

- json-schema.org & jsonschema.net
- jsonapi

测试框架工具

- Karma
- Intern
- NightWatch.js

前端数据存储工具

- YDN-DB
- forerunner
- AlaSQL
- LokiJS
- lovefiled
- Dexie.js
- localForage
- pouchdb

模块/包加载工具

- SystemJS
- webpack
- Broeserify
- rollup.js

模块/包仓库工具

- NPM
- Bower
- jspm.io
- spmjs

Web/云/静态主机托管工具

- AWS [$]
- Heroku [free to $]
- DigitalOcean [$]
- Modulus [$]
- DIVSHOT [free to $]
- netlify [free to $]
- surge [free to $]

项目管理 & 代码托管

- Github [free to $]
- Codebase [$]
- Bitbucket [free to $]
- Unfuddle [$]
- Assembla [free to $]

合作 & 交流工具

- Slack & screenhero [free to $]
- Skype [free to $]
- Google Hangouts

代码**/Github** 合作 **&** 交流**:**

- Gitter [free to $]

译者补充**:**

- Gitup

内容管理托管/API工具

API CMS 工具:

- prismic.io [free to $]
- contentful [$]
- Cosmic JS [free to $]

Hosted CMS tools:

- LightCMS [$]
- Surreal CMS [$]
- Page Lime [$]
- Cushy CMS [free to $]

Static CMS tools:

- webhook.com

前端开发者的后端服务工具(又称 BASS: Back-end as a service)

数据/后端服务管理:

- Firebase [free to $]
- Parse [free to $]
- kinvey [free to $]
- Back& [free to $]
- Pusher

用户服务管理:

- UserApp [free to $]
- hull [$]
- auth0 [$]

离线工具

- upup
- offline.js
- pouchdb
- hood.ie

安全工具

Coding tool:

- DOMPurity
- XSS

References:

- HTML5 Security Cheatsheet

任务管理(又称 构建)工具

任务管理/构建 工具:

- Gulp
- Grunt

Tasking/build and more tools:

- Brunch
- Mimosa

部署工具

- FTPLOY [free to $]
- Travis CI [free to $]
- codeship [free to $]
- Bamboo [$]
- Springloops [free to $]
- surge
- sync ninja

网站/APP 监控工具

Uptime:

- pingdom [free to $]
- Uptime Robot
- Uptrends [$]

General:

- New Relic

JavaScript 错误监控工具

- Raygun [$]
- errorception [$]
- sentry [free to $]
- {track:js} [$]

性能工具

Reporting:

- WEIGHTOF.IT
- Web Page Test
- GTmetrix
- Speed Curve [$]
- Chrome Devtools Timeline
- sitespeed.io

JS tools:

- ImageOptim-CLI
- imagemin

Budgeting:

- performancebudget.io

SVG 工具

优化:

- SVGOMG
- Peter Collingridge's SVG Optimiser
- SVGO
- SVGO-GUI
- SVG Cleaner
- Scour SVG Scrubber
- Clean Multiple Gradient Stops

SVG 编辑器:

- Illustrator
- Sketch
- Inkscape
- DrawSVG

创建雪碧图:

- Icomoon
- Fontastic
- Grunticon

Bug 追踪:

- SVG Weirdness
- SVG Edit